PCT WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 17/30	A1	(11) International Publication Number: WO 00/63801
		(43) International Publication Date: 26 October 2000 (26.10.00)

(21) International Application Number: PCT/US00/08744

(22) International Filing Date: 21 April 2000 (21.04.00)

(30) Priority Data:
60/130,321 21 April 1999 (21.04.99) US

(71) Applicant *(for all designated States except US)*: TONI DATA, LLC [US/US]; 2666 Airport Road South, Naples, FL 34112 (US).

(72) Inventor; and
(75) Inventor/Applicant *(for US only)*: RAMLEY, Intesar, F., K. [CA/CA]; 1991 W. 62nd Avenue, Vancouver, British Columbia V6P 2G5 (CA).

(74) Agents: NEILS, Paul, F. et al.; Sughrue Mion Zinn Macpeak & Seas, PLLC, Suite 800, 2100 Pennsylvania Ave., N.W., Washington, DC 20037-3213 (US).

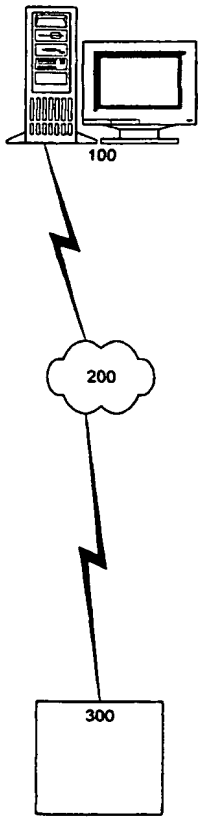
(81) Designated States: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW, ARIPO patent (GH, GM, KE, LS, MW, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

Published
With international search report.
Before the expiration of the time limit for amending the claims and to be republished in the event of the receipt of amendments.

(54) Title: MANAGED REMOTE VIRTUAL MASS STORAGE FOR CLIENT DATA TERMINAL

(57) Abstract

A client data terminal (100) subscribes to a managed data parking service and stores file. A virtual hard drive includes remote files stored at the server (300) and local files stored at the client data terminal (100). A virtual hard drive agent at the terminal (100) provides a user interface whereby a user may put files to the server (300) or get them to the terminal (300), with memory management being automatically handled. Backup status information controls local file deletion activities. Synchronization operations ensure more recent file versions are stored.



FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

MANAGED REMOTE VIRTUAL MASS STORAGE FOR CLIENT DATA TERMINAL

CROSS-REFERENCE TO RELATED APPLICATIONS.

This application claims the benefit of U.S. Provisional Application No. 60/130,321, filed April 21, 1999.

BACKGROUND OF THE INVENTION.

Field of the invention.

This invention pertains to the field of mass storage management for a computer. More particularly, this invention pertains to a client-server system for supporting remote mass storage on a server for a client computer system. The system of the invention is applicable to any client-server system, but is particularly advantageous for use in a system having a handheld computers. A system according to the invention may be thought of as providing a managed data parking service (MDPS) for clients, wherein the parking of the data is managed by the server.

Related work.

Handheld computers are known for being lightweight and small. To achieve these advantages, handheld computers have no mass storage units. The lack of mass storage for a handheld computer is a drawback when additional storage is required.

It is possible to carry around a mass storage unit such as a disk drive or the like, but this detracts from the enjoyment of a user in having a lightweight, small, handheld computer. Such a solution also usually requires electrical power, or at least increases the demand on battery power.

What is needed to improve handheld computer functionality is mass storage without extra hardware or increased electrical power requirements. Non-handheld computers would also benefit from such a solution as well.

SUMMARY OF THE INVENTION.

It is an object of the invention to provide remote mass storage, for handheld and other computers, without adding weight or increased electrical power consumption. It is a further object of the invention to make the access to and use of the remote mass storage very simple and substantially transparent to the end user.

To achieve these objects and to provide several additional advantages which shall become apparent from the discussion below, the invention provides, in a presently preferred embodiment, a system, method, computer program product, and user interface for managed remote virtual mass storage. The invention is applicable to all types of computers, but particularly advantageous in computers having limited mass storage or no mass storage. The invention is especially advantageous for handheld computers.

According to an embodiment of the invention, there is provided an integrated data management system (IDMS). The IDMS has a higher level system management tier and lower level individual computer management tier. The higher level management tier includes the software systems that provide managed data parking, network management, system management,

security, and Internet services. The lower level includes a software agent set that resides inside the individual computer. The software agent set advantageously has a design independent of the operational environment of any particular individual computer.

In a preferred embodiment, the software agent set includes a master agent adapted for communication with an external system manager of the higher level tier so that the system manager can exercise control of sub-agents in the software agent set. The sub-agents include a virtual hard drive sub agent supporting data parking, a status sub-agent reporting hardware and software changes to the system manager, and a software inventory sub-agent delivering a current software applications inventory based on a subscription status of the user.

The subscription status of the user and other user information is kept at the higher level tier. The software agent set stores data in a mass storage at an external location, such as a server, as determined by the system manager.

In another preferred embodiment, the software agent set includes a sub-agent for setting an underlying management information base, and a security sub-agent supplying logon parameters to the security system at the higher level tier. The higher level tier refuses to provide any stored data from the external location in the event the user of the computer does not provide proper authentication.

By virtue of the cooperation between the higher and lower level tiers, a user of a handheld computer can store and retrieve files from mass storage without carrying about any additional equipment, and without increased electrical power requirements. The security features prevent a thief of the handheld computer from making use of any external data.

Additional objects and advantages of the invention will become apparent from the detailed description below, taken in conjunction with the appended drawing figures.

BRIEF DESCRIPTION OF THE DRAWING FIGURES.

Fig. 1 shows a client and a server in communication through a network.

Fig. 2 shows, in a highly simplified schematic form, a server arrangement of modules.

Fig. 3 shows, in highly simplified schematic form, a handheld computer.

Fig. 4 shows a set of software applications constituting a software agent set of the handheld computer.

Fig. 5 shows an example window for a user interface.

Fig. 6 shows the example window from Fig. 5 with some added detail and an example of data as it might be displayed.

Fig. 7 shows a flowchart relating to a set up operation for a managed data parking service.

Fig. 8 shows a flowchart relating to another set up operation for the managed data parking service.

Fig. 9 shows a flowchart relating to operations for adding a box to a virtual hard drive.

Fig. 10 shows a flowchart relating to synchronization operations and the harmonization of file and directory structures.

Fig. 11 shows a flowchart relating to a put operation for sending a file from local to remote storage.

Fig. 12 shows a flowchart relating to a get operation for retrieving a file from remote to local storage.

Fig. 13 shows a flowchart giving a higher level view of a synchronization operation.

Fig. 14 shows a flowchart for explaining a memory management approach used during a get operation.

Fig. 15 shows a flowchart relating to file deletion operations.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS.

Glossary.

This section provides some definitions of terms that may be used throughout this description.

Agent: An intermediary, or proxy between a management application within a boundary of a managed domain and a managed DT (see DT definition), acting upon managed DT on behalf of a management application. The agent software exchanges management information with management application using a network management protocol (e.g. TCP/IP).

Client Data Terminal (CDT): DT (see definition) in a client/server configuration. The CDT can be a node within a managed domain. In such a case the CDT has the necessary set of software agents which facilitates the management function.

Cold Start: The process of bringing the system into an operational state from cold hardware state: either a powered-off state or a boot-ROM active state. The boot-ROM state is the state of a halted DT state prior to being powered-off.

Data Terminal (DT): Any node (device) or a node within a network which is processor based, e.g. Handheld PC, desktop PC, server, or any mobile data host such as an organizer, smart cellular phone, personal digital assistant, or the like.

Domain: Used in the context of management to mean the extent or scope of full or partial control.

Infrastructure Software: Software that forms a fundamental underlying layer of the system domain.

Initialization data: An umbrella term for the data used by the system when starting up to provide itself an initial operational context.

Last State Data (LSD): Data saved by the system during operation, to be used in the event of failure to restore the system to an up-to-date state. It is the duty of the management application to collect the necessary status quo of all client data terminals and store them in a specified file (LSDF).

Managed Client Data Terminal (MCDT): CDT in a managed environment; may be referred to simply as "terminal" throughout this description. If

Managed Object: Managed node/element participating or not participating within a managed network in a managed domain. Also a managed object can be a class type object or application object which has its own published interface through which the management-application can manipulate the underlying object.

Management Application: The software, normally resident on a management station that provides the management functions. Management is achieved via bi-directional exchange of management information with agents using the network management protocol.

Management Station: A particular node within a system with domain management responsibility; because it has a management application.

Mass Storage: It is a medium where data can be stored in mass. The mass storage device can be in the form of Hard Drive (HD) or write/read CD-ROM, or any other similar device for storing a large amount of digital information.

Offline: An equipment state indicating that the equipment is not participating in overall system operation.

Online: An equipment state indicating that the equipment is ready and selected to participate in overall system operation.

Platform Shutdown: A process of collectively halting equipment after a system shutdown has been performed.

Platform Startup: An initial phase of a system cold start, involving the collective powering-on of equipment to bring it online prior to starting system operation; required after a total system shutdown involving a power-down or after a total system failure involving loss-of-power.

Software Agent Set: A set of software agents on a terminal.

Software configuration : The organization of software resources which provides support for a particular system or collection of systems operation.

System Configuration : The physical plus adapted behavior of a terminal, including initiation or continuation data residing in a resource file, to provide a particular equipment configuration and software configuration based on target behavior in order to achieve a specified system performance.

System Management: Management of an overall system including, but not limited to, Configuration/State management, Fault management, Performance management, Security management, and System administration.

System Shutdown: The process of shutting-down all system functions; transitioning the system from an operational state to a designated state (e.g. maintenance mode state).

System Start-over: The process of bringing the system into an operational state (such as full service mode, reduced capability mode, normal mode, or standalone mode) from the state that is based on or initialized from saved data.

System Startup: The process of bringing the system into a clean operational state (full service mode, reduced capability mode, normal mode, or standalone mode) from the any system state.

An embodiment.

The invention will now be described by way of a non-limiting example.

In Fig. 1, there is shown a server 100, a terminal 300, and a network 200 interconnecting the two. The server 100 provides a managed data parking service (MDPS) for the terminal 300.

The terminal 300 is a terminal used by a user, and may be referred to as a client data terminal (CDT). The terminal may be of any type, such as a PC or the like, but is ideally a handheld computer (HHC). For the remainder of this example, it will be assumed that the client data terminal is a handheld computer, but it will be understood that the example applies equally to other types of computing equipment that are not hand held.

Fig. 2 shows, in highly simplified schematic form, server 100. Server 100 includes a server database system 105 (DB), a server system manager module 110 (SMM), a server network monitor module 115 (NMM), a server managed data parking module 120 (MDPM), a server security module 125 (SECM), a server request exchange module 130 (REM), a server proxy module 135 (PM), and a server internet information module 140 (IIM).

The server database system 105 may be a relational database. The server database system serves as a collection of repositories for all activities involving the service. It serves the system reporting engine, a subscription system, a billing system, and order process system, and inventory system, remote virtual hard drive activities and usage, user profiles, and the like.

The server system manager module 110 provides a wide range of management support for the managed data parking system and the terminals 300. One of the functions of the server system manager model 110 is to communicate with a master agent on the terminal 300, to be described later. The purpose of this communication between the server system manager module 110 and the terminal master agent is to provide for the management of all of the other agents on the terminal 300. By virtue of this control of the server system management module exercised over the agents on the terminal 300, the server system manager module can manage (a) all aspects of the hardware and software of the terminals 300 and networks, as necessary; (b) the overall configuration of the entire managed domain; (c) the overall domain startup, start-over, and shut down operations; (d) the starting, shut down, and change of state for one or more terminals 300 within the managed domain as necessary; (e) maintaining the integrity of the overall domain and its participant terminals 300; (f) a server system manager module log file for reporting; (g) other log files as necessary for the

generation of reports; (h) storing the system's status in an LSDF; and (i) a software interface with the server database system 105 enabling a system operator to control a motive operation of the managed data parking system.

The server system manager module 110 may include a graphical user interface. Such a graphical user interface may be provided so that a systems operator, using the interface, can cause the execution of certain functions of the agents on a terminal 300 so as to start, shut down, or restart a participant terminal 300. The graphical user interface which may be provided with the server system manager module 110 may also permit a system operator to view the status of segments of the system's overall domain.

The server network monitor module 115 monitors the network 200 through the server internet information module 140. The server network monitor module 115 may provide alerts and other notifications to the server system manager module 110. The server internet information module 140, of course, need not operate on only the internet but is a representative module that provides an interface with whatever network the server 100 is connected to.

The server security module 125 provides the functions of checking the security information (i.e., encryption format, initialization, call, authentication, authorization, and other security components) coming from a terminal security agent on the terminal 300 (to be described below). The server security

module 125 also maintains security files in the database 105, and also is responsible for identifying the registered location of an active terminal 300. This is one of the necessary security measures to validate the legitimacy of the terminal 300 participation in the managed domain. The server security module 125 may carry out its functions by verifying logon passwords and also by checking the subscription status of a user in a subscription verification table stored in the server database system 105.

The function of the server request exchange module 130 is the routing to the right destination on the server side of any request coming from one of the terminals 300. This module recognizes the validity of the request type and passes it to the security system.

The function of the server proxy module 135 is to act as a typical proxy server.

Although the parts of the server 100 have been described as modules, it will be appreciated that they may also be referred to as servers (i.e., database server, proxy server, request exchange server, and the like) or as applications. The modules may be hosted on only one or on several computer systems. The modules may be implemented in any of a variety of ways, using custom designed components with commercial products, using any of a variety of programming paradigms.

The server managed data parking module 130 is a novel component on the server 100 designed to provide management for

the remote virtual hard drive (RVHD) system and to manage the handling of the terminals 300 validated, verified, prioritized and time-scheduled requests. These requests are for user business and/or computing data storage or retrieval. The data can be text, graphics, text and graphics, executable binaries or any other format of data. The managed data parking service realized via the server managed data parking module 130 provides the management (i.e., monitoring and control) for RVHDs with security accessed, independent, mapped, and totally isolated hard drive segments. The management functions of the server managed data parking module 130 include: (a) providing access security to directories and files (security accessed units); (b) user/directory boundaries allocation (mapping); (c) data compression and data distribution over the RVHD segments (totally isolated hard drive segments); (d) health status of the virtual hard drive and its segments; (e) activity (event and error) logging (monitoring); and (f) support for intelligent bi-directional data transport in collaboration with a virtual hard drive agent on the terminal 300. This function (f) is achieved via multithreading, which enables server managed data parking module 130 to allocate different object instances each within a memory thread sector.

This server managed data parking module 130 may have its own graphical user interface to permit operator intervention. The interface of the server managed data parking module 130 may be referred to as a managed data parking system monitoring

system (MDPS-MS). The MDPS-MS allows the operator to view the health status of the segments of the mass storage which is managed on the server side. MDPS-MS also displays the logging activity.

The preferred embodiment provides for two modes of operation: manual and automatic. The two modes apply to data compression and data distribution. A system operator can control the two modes through the MDPS-MS.

Requests related to data parking coming from the terminal 300 originate from a virtual hard drive agent. The server validates each request to ensure that the request is within the allowed size. The server verifies each request to be sure that the storing, synchronization, backup, or retrieval is performed in a correct manner. The server also answers the request within the priority and time schedule.

The terminal 300 is shown in Fig. 3. Also shown in Fig. 3 is a software agent set 305 (SAS); a terminal operating system 310 (OS); a terminal kernel 315 (K); a terminal display hardware 320 (DHW); a terminal input/output system 325 (IO); and terminal communications hardware 330 (CHW). The terminal communications hardware 330 is shown in communication with network 200.

Fig. 4 shows a more detailed view of software agent set 305.

The software agent set 305 includes a terminal master agent 305-1 (MA); a terminal virtual hard drive agent 305-2 (VHDA); a

terminal security agent 305-3 (SECA); a terminal status agent 305-4 (STA); and a terminal user data agent 305-5 (UDA).

In a system according to the preferred embodiment, the participant CDTs are managed using the software agent set 305. The software agent set 305 uses a management / collection model. More precisely, the storage of the participants is managed in a highly automated fashion using the software agent set 305. In this model the collection is a container which represents the terminal master agent 305-1. The terminal master agent 305-1 is the only agent on the terminal 300 that communicates with the server system manager module 110. This communication may be based on a protocol on top of TCP/IP (see discussion of capsules, below). To make an effective use of the communication medium bandwidth, both the terminal virtual hard drive agent 305-2 and server system manager module 110 cause a convolvement of the data through a compression algorithm. The server 100 is the issuer of this algorithm. Compression algorithms are well known in the art, and one of skill in this field will be capable of selecting an appropriate compression algorithm.

It will be appreciated, therefore, that there is a management connection between the server system manager module 110 of server 100 and the terminal master agent 305-1 of the software agent set 305 of the terminal 100. This management connection is understood to take place via the network 200. As will be understood by those familiar with modern data communications, the network 200 may be the internet (i.e., a

worldwide public network), an intranet, any other type of public or private network, including even the public telephone network and wireless networks such as cellular or satellite networks. The key point is that the server 100 be able to communicate with the terminal 300 (i.e., with the participating CDT's).

Each CDT can, of course, operate on its own during a time in which there is no management connection or during the time in which the management connection is intentionally suspended. Such independent operation allows the terminal 300 to interact directly with the network for other purposes. For example, when the network 200 is the internet, the terminal 300 operating in an independent mode may be used for browsing the World Wide Web, executing file transfer tasks, or the like. In the independent mode, the terminal 300 may also be used to perform tasks supported by the system platform.

Terminal virtual hard drive agent.

The terminal virtual hard drive agent 305-2 resides inside the terminal 300. The terminal virtual hard drive agent 305-2 is responsible for providing intelligent bi-directional support for the terminal 300 side of the system.

The terminal virtual hard drive agent 305-2 may be thought of, in an actual implementation, as being provided in versions falling into two main categories. A browser based version and an operating system environment dependent version. The latter category describes versions of the terminal virtual hard drive agent 305-2 tailored to particular operating systems (terminal

operating system 310) such as Win CE, Windows 95/98, Windows NT, OS, Mac, Linux, Unix, and all notebook environments. All versions start from the same base software but they might differ in their GUI layouts.

To fulfill the above and other functions, the terminal virtual hard drive agent 305-2 obtains, determines, and/or keeps certain information.

The information with which the terminal virtual hard drive agent 305-2 is concerned includes information about the current net storage capacity of the terminal 300. The current net storage capacity of the terminal 300 may be determined by a direct call to the operating system. For Win CE, for example, the storage capacity is the actual dynamic memory space. For Win 95/98 and NT, the current net storage capacity is the assigned destination hard drive. The determination of the current net storage capacity is also possible for other operating systems such as OS, Mac, Linux and Unix. This functionality allows the terminal virtual hard drive agent 305-2 to allow the user to set a maximum mass-storage threshold so as to protect the storage medium against data flooding.

To accomplish this, the terminal virtual hard drive agent 305-2 may periodically check the current net storage capacity. When the current net storage capacity indicates that the maximum mass storage threshold has been met, then the terminal virtual hard drive agent 305-2 may undertake to read a predetermined or previously stored file movement policy. Based on the file

movement policy, the terminal virtual hard drive agent 305-2 may make a limited/conditional file list of files. Given the condition that the total size of files should be under the maximum mass-storage threshold, the terminal virtual hard drive agent 305-2 can iterate through this list; select files for movement from local mass-storage to an assigned destination; move the selected files and update status information.

The terminal virtual hard drive agent 305-2 is also concerned with the actual file destination after transport. The file actual destination after transport, by default, will be the server 100. In the event that the system is implemented within a defined local domain (such as a LAN within a company), then the destination can be any mass storage media located within the host domain.

The terminal virtual hard drive agent 305-2 is also concerned with the actual transportation time of files. When a user has a long list of files to be transported, the terminal virtual hard drive agent 305-2 permits the user to set the date, time, and action type. In case there is no such user-prepared list, then the terminal virtual hard drive agent 305-2 may take certain predetermined steps, beginning with checking the availability of the management connection.

When the management connection is not available, no action can be taken with respect to a file. When the management connection is available, the requested or necessary actions can

immediately be undertaken and the status information updated appropriately.

The terminal virtual hard drive agent 305-2 may provide a user interface that allows the user to schedule tasks, and also to review the scheduled tasks for viewing and editing.

As mentioned above, communication between the server system manager module 110 and the terminal master agent 305-1 may include the provision of a special protocol to convey file attributes. The special protocol is preferably realized as a data capsule of certain information.

Another type of information with which the terminal virtual hard drive agent 305-2 is concerned is the file attributes of the files. The data capsule of the special communication protocol on top of the TCP/IP layer contains an extended set of attributes for each file. This extended set of attributes includes, but is not limited to, the following:

- full name;
- size;
- type;
- location;
- last saving date; and
- generic set of attributes (for example, whether the file has a backup attribute set, scheduled task set, and the like).

A mass-storage related request from the terminal master agent 305-1 of the terminal 300 to the server system manager module 110 of the server 100 preferably has a request priority.

The priority of a request is another type of information with which the terminal virtual hard drive agent 305-2 is concerned.

The MDPS system has defined therein an internal priority set. The terminal virtual hard drive agent 305-2 implements this internal priority set so as to prioritize a given mass-storage related request at both the terminal 300 and server 100 sides. This set is supplied by the MDPS and presented by the terminal virtual hard drive agent 305-2 to the user.

The terminal virtual hard drive agent 305-2 also is concerned with the last file write/read position before a communication interruption. This parameter is stored by the terminal virtual hard drive agent 305-2 on the terminal 300 when the action mentioned in a request message is a "get" action (discussed more below). Otherwise, this parameter is stored on the server 100. Also, the terminal virtual hard drive agent 305-2 is concerned with whether a given request is a new request or is a request previously interrupted. To this end, the terminal virtual hard drive agent 305-2 keeps a flag which may be referred to as a continue flag. Thus, when a request is sent to the server 100 for data transport, it is sent with the continue flag indicating "new" or "continuation".

In the preferred embodiment, the user has the ability to set several options with respect to the terminal virtual hard drive agent 305-2, as summarized in the following table.

File synchronization by default	Yes/No
Minimum Space for local storage	------ Mbit or Kbit
Actions on files – Move, Comprise or both o Move By policy New Files, Old Files or Specified o Comprise By policy Default or Defined	
Establish server link by default	Yes/No
Move all local files to server on logoff	Yes/No
Bring newest file from server	Yes/No
Enable Versioning for all files	Yes/No
Allow Multi-users for all files	Yes/No

Terminal security agent.

The terminal security agent 305-3 resides on the terminal 300 and is in the software agent set 305. This agent supplies the terminal 300 logon and subscribed services security parameters to the server security module 125 of the server 100.

Terminal status agent.

The terminal status agent 305-4 resides on the terminal 300 as part of the software agent set 305. The terminal status agent 305-4 reports any changes in the terminal 300 hardware and software status to the server 100. The terminal status agent 305-4 triggers the execution of the installing or the removing of software versions, depending on the reported software inventory supplied by the server 100.

Terminal user data agent.

The terminal user data agent 305-5 resides on the terminal 300 as part of the software agent set 305. The terminal user data agent 305-5 keeps up to date user related data (which covers user subscription/logon detail, the maximum authorized storage size, number of existing files, last communication

detail, CDT unique identification number, etc...). This information is delivered to the CDT. Each component of this information is required for a particular security issue. Some of these info can be displayed in order to enable the user to know the status quo of MDPS usage.

Terminal master agent.

The terminal master agent 305-1 may be thought of as an agent manager for the software agent set 305. The terminal master agent 305-1 is an application which has a GUI to allow the terminal 300 user to see the current software agent set 305 activities and set the operational options. The agent manager allows the user to exercise some or overall control over software agent set 305 activities.

Operations.

In a preferred embodiment, a user must subscribe to receive the managed data parking service, and the subscription must be approved. Users may subscribe by any conventional way such as completing a form on a web site, by fax, by telephone, or the like. The user may be requested to provide user related information such as the hardware type of the terminal 300, the host domain IP address and/or name of the terminal 300, the terminal 300 operational environment (i.e., operating system, memory, processor speed, connection, and the like), user personal information, and the particular services desired by the user to include the maximum mass storage being requested by the user.

This information can be the basis for a billing and security system. Invoicing and payments may be advantageously handled through a web site of the service, and also through the more conventional means such as mail.

After beginning a service subscription, the user with the terminal 300 is registered for managed data parking services with the service provider. The terminal 300 of the user is in communication with the service via an internet line of the required bandwidth or via a dialup line connection to a modem bank.

In addition to being registered with the service provider, the user has also downloaded the software agent set 305. The software agent set 305 may be said to include a graphical user interface constituting a mobile explorer. In an embodiment, the mobile explorer is a feature of the terminal virtual hard drive agent 305-2.

Although the agents of the software agent set have been described as "agents", it will be appreciated that these may be implemented as processes or routines, and some of them could be combined based on the implementation decision. It will also be understood that the agents could exist as objects in an object-oriented system, and the downloading of the software agent set could include simply the downloading of an appropriate software application together with an object class library.

When the user registers for the service, it is also the case that a configuration file is downloaded and stored on the

terminal 300. It will be appreciated that the configuration file contains information relating to the user's registration with the service, such as whether the user has arranged for normal security or for higher security for his files, platform information, maximum space rented from the service, and the like.

The user, once registered, must set up his virtual hard drives. The term virtual hard drive (VHD) is used to embody the concept that the terminal 300 appears to have a hard drive that it does not have on board. This term can also be applied to a client that is more full-featured than a terminal 300, even though such a client may have its own hard drive, because the system and method of the invention provides for such a client a hard drive capability beyond that which would normally be available.

A user may set up more than one VHD. For the sake of simplicity, for the remainder of the explanation, it will be assumed that the user has set up only one VHD.

Returning to the example of the terminal 300, let it be assumed that the VHD set up by the user is designated by the user as "virt-d". The VHD could be given a letter name, as well, such as "V:".

The concept of a box will now be explained. A box may be thought of as a super-folder. When a user subscribes to the virtual hard drive service, a portion of storage on the server is a set aside for that user. That storage is the user's

virtual hard drive and is defined as a mapped disk. Within the virtual hard drive is one or more boxes. A box may be thought of as a convenient way of grouping files. A box also provides a convenient way for a user to add files to the virtual hard drive so that they can, from that point on, be managed by the managed data parking service. A box can have sub-folders. A box also can have certain attributes that are effective for each of the files in the box or in any of the sub-folders of the box. For example, a box may have the "backup" attribute set, in which case each file within that box will also have the backup attribute set automatically without the need for setting the attribute for each individual file. The backup attribute is described more below. When a user registers, a default box is defined for him.

Given that the user is registered, has the software agent set 305, and has set up his VHD virt-d, the operations of Put and Get will now be discussed. First, however, a few terms will be defined. Herein, the term "local" refers to the terminal 300. "Remote" refers to the server 100 providing data parking service. Likewise, "download" means to move data to the terminal 300 and "upload" means to move data to the server 100.

Assume that when the client is turned on, the user decides he wishes to create a new report or other file. The user creates the file using an application, such as a word processing application or the like. The user decides it is desired to save the file to the virtual hard drive.

To carry out such an operation, the terminal virtual hard drive agent 305-2 must be active. Activation of the terminal virtual hard drive agent 305-2 can be completely manual. This is not preferable since one of the objects of the system according to the invention is to provide a convenient and automatic managed data parking service. Another option is to cause activation of the terminal virtual hard drive agent 305-2 upon startup of the terminal 300. This is advantageous in that the user need never be troubled with discerning whether the terminal virtual hard drive agent 305-2 has been started or not. Another option is to activate the terminal virtual hard drive agent 305-2 only in response to a request for a file or the like.

The terminal virtual hard drive agent 305-2 may be used to display the mobile explorer window. Such a window is for navigating a tree-shaped directory structure of files on the drive virt-d. The mobile explorer provides such conventional file information as name, size, type, and date. The mobile explorer also provides additional information relating to whether a file is identified for backup. The mobile explorer further provides additional information relating to whether a file is presently local or remote.

The user desires to store his newly-created file on the VHD virt-d. The user uses the mobile explorer to indicate that the file is to be stored remotely.

An important feature of the invention is the manner in which files are brought from the server 100 to the terminal 300 and also the manner in which files are backed up from the terminal 300 to the server 100. When a user wishes to have a file copied from the terminal 300 to the server 100, the operation is known as a "put" operation. When a user wishes to have a file from the server 100 made available on his terminal 300, the operation is a "get" operation. It will be appreciated that the operations can also be described as the putting and getting of files.

Let it be assumed that the terminal 300 has a certain amount of memory, and in this memory there is a plurality of files that have already been gotten from the server 100. When the user decides to request another file, and when the size of the file will not fit within the available memory, certain steps must be undertaken.

The way that this is handled according to a presently preferred embodiment of the invention is as follows. The files already in memory in the terminal 300 are inspected. A determination is made to find out which of the files is the least recently used file. If the least recently used file is identified and is automatically backed up to the server 100. Once the file that was least recently used is successfully backed up to the server 100, it is deleted from the memory of the terminal 300. When the file is deleted from the terminal 300, the location of the file is changed from "local" to

"remote". This change in status indicates to the user that the file is no longer resident at the terminal 300. This operation to back up the least recently used file is undertaken whether or not the file has been given a status of "back-up" or not.

The reason that this operation is undertaken regardless of the "back-up" status is that the file might still be being worked on by the user, and it would be inappropriate to delete this file without keeping a copy.

When the least recently used file is deleted from the memory of the terminal 300, a second determination is made. The second determination involves determining whether there is enough available memory to accommodate the file that the user is seeking to obtain by using the "get" operation.

When the second determination indicates that sufficient memory is available, the get operation proceeds. When the second determination indicates that the memory is still insufficient to accommodate the desired file, the files presently in memory are again inspected to determine which one is the least recently used file.

The least recently used file is again backed up to the server 100, and deleted from the memory of the terminal 300. This process of inspection, deletion, and subsequent determination as to sufficient memory space is repeatedly undertaken until there is enough memory space to accommodate the desired file.

There are occasions in which a user may request a file that is too large for the terminal 300 to accommodate even when there are no other files present in the memory of the terminal 300. In such a case, there are two preferred ways of handling this.

According to one embodiment of the invention, an error message is produced, indicating to the user that the size of the file that has been indicated by the request for the "get" operation is too large, and that it is impossible to get the file.

In another embodiment, a paging algorithm is undertaken to provide for the user the desired portions of the file. In particular, such a paging operation might involve obtaining the first part of the large file. If the user performs an operation that requires a portion of the file that is not present, then the first part of the file, or a subset thereof, is copied back to the server 100 and a subsequent portion of the file is retrieved. Various paging algorithms are well-known in the computer art, and it will be appreciated that the nature of the user applications may dictate which paging algorithm is the best to use.

The "put" operation is much simpler because it is not possible for a file on the terminal 300 to be larger than the memory of the server 100. When the user requests a "put" operation, the file is simply copied back to the server 100.

Sometimes, the user may request that a file on the terminal 300 be put to the server 100, but the version on the server 100

is just as current as the version in the terminal 300. Whether this is the case can be verified by checking the modification date and time of the file on the terminal 300 against the modification date and time of the file on the server 100. When the two files have identical information, then there is no need to put the file from the terminal 300 to the server 100. In such a case, the program does not actually communicate the file to the server 100. Rather, the interface on the terminal 300 gives an indication that the back-up operation is complete, marks the status of the file as "remote", thus giving the appearance to the user that the "put" operation was carried out.

In an alternative embodiment, it is possible to prompt the user with the modification dates and times of the terminal 300 version and the server 100 version of the file, and confirm whether the user wishes actually to replace the server version of the file with the terminal 300 version of the file. This embodiment is advantageous because it gives the user control over whether to actually copy over the server version of the file with the terminal 300 version of the file.

A description of the mobile explorer window will now be given with reference to Fig. 5. In Fig. 5, reference numeral 500 indicates the explorer window. The explorer window 500 includes a split screen display with a mobile explorer file tree area 510 and a mobile explorer file listing area 520. The mobile explorer window 500 also includes a mobile explorer toolbar 530 which may have a conventional menu including

operations such as "file", "view", "action", and "help". The mobile explorer toolbar 530 may also include some buttons so as to make certain commands available in a convenient manner. The buttons may be thought of as shortcuts for selected ones of the commands which may be accessed through the menu.

Fig. 6 shows the mobile explorer window 500 with some representative information included in the mobile explorer file tree area 510 and the mobile explorer file listing area 520. The mobile explorer toolbar 530 includes a mobile explorer put button 540, a mobile explorer get button 550, and a mobile explorer synchronization button 560.

The mobile explorer put button 540 begins an operation to put one or more selected files from the terminal 300 to the storage on the server 100. This operation could be invoked in numerous other ways, such as by accessing a put commend through the menu in the mobile explorer toolbar 530 or by pressing a certain button on the terminal 300, or even by giving an appropriate voice command when the terminal 300 is adapted to respond to voice commands. Furthermore, the mobile explorer put button 540 could itself be activated by an operation with a pointing device (e.g., the click of a mouse, tap of a glide pad, touch of a stylus, or the like).

It will be appreciated that, in this description, a mention of "clicking the mobile explorer put button 540" is used as a linguistically convenient way of expressing the invoking or

activation of a put operation. The same applies to the other operations mentioned herein.

The mobile explorer get button 550 is for activating a get operation to obtain one or more files from the server 100

The mobile explorer synchronization button 560 activates a synchronization operation between files local to the terminal 300 and files remotely stored at the server 100.

In the mobile explorer tree area 510, there is shown a tree-like structure used to provide an intuitive display of the file system of the user. The virtual hard drive defined for the user is VHD. The hard drive is hosted on the server 100, but appears to be a folder on the terminal 300 of the user. The display of the tree-like structure shown in Fig. 6 is conventional, and any other way of showing a tree-like structure would be acceptable. Furthermore, the split pane view is not necessary to the practice of the system described herein, but is a helpful adjunct to facilitate easy user operation and understanding.

In the representative display shown in Fig. 6, the virtual hard drive has one sub-folder FA. The folder FA may be understood to be a box, as described above. In the display, the icon for FA could be represented in many alternative ways, such as a three-dimensional box. Since a box is substantially a super-folder, FA will be described as being a folder for the remainder of this discussion.

To the left of the icon representing VHD is an expansion / contraction symbol which may show a "-" when the subfolders of a given folder are displayed, and which may show a "+" when the subfolders of a given folder are not displayed. In the present example, the folder VHD is shown in expanded form, and its single subfolder, FA, is shown.

The folder FA is shown also as being in expanded form, and it has two subfolders, viz., FB and FC. The folder FA is shown with an arrow next to it indicating that folder FA is the folder whose contents are being displayed in the mobile explorer file listing area 520. The folder FA may be thought of as the current folder or current directory.

The mobile explorer file listing area 520 shows the names of any subfolders in the current folder, as well as the names and details concerning any files therein. In the representative example shown in Fig. 6, the mobile explorer file listing area 520 shows subfolders FB and FC, as well as four documents named "Doc 1", "Doc 2", "Doc 3", and "Doc 4".

For each of the documents in the current folder there may be displayed certain additional, detailed information such as the size of the file (S1, S2, S3, and S4); the date and/or time of modification (T1, T2, T3, and T4). There may also be shown status information indicating the virtual status of the documents.

The document Doc1 has a status of "L", which represents that the file exists locally in the memory of the terminal 300.

The document Doc2 has a status of "R", which represents that the file exists remotely at the server 100 and not at the terminal 300. The document Doc3 has a status of "E". This status means that Doc3 has just been removed from control of the minutes to data parking service. That is, just as a user can include files under the management of the managed data parking service, a user can also withdraw files from this control and manage them as they normally would in the absence of such a system.

The document Doc4 has a status of "N" meaning that this document needs to be backed up to the server 100. This status may indicate that the user has created of file within the domain of the managed data parking service, but that the file has not yet ever been put to the server or that a put attempt has been made without success and the file still needs to be backed up to the server.

Setup.

Fig. 7 shows a flowchart for an operation and which the user specifies certain parameter settings. These parameter settings provide the details of the server to which the terminal 300 is to connect with when synchronizing a virtual hard drive. It is preferred that such settings have default values.

It is assumed that the user has already subscribed to the service, and installed the terminal virtual hard drive agent 305-2 and the rest of the software agent set 305 on the terminal 300.

In step 700, the user starts the mobile explorer. In step 710, the user employs menus or buttons as necessary to initiate the appropriate command. In step 720, the mobile explorer responds to the activation by displaying a connection settings dialog box showing the current values. In step 730, the user modifies the connection settings as desired, and initiates an action to update the connection settings. The updated connection settings are stored by the mobile explorer in step 740. With the connection settings adjusted as desired by the user, the user can use the terminal 300 to connect through the network 200 to the server 100.

Another aspect of setting up the service is the setting up of the user's box.

Fig. 8 shows the setting of options related to user boxes.

In step 800, the user, who already has started the mobile explorer, activates a box options operation to update and/or enter options relating to the user box. In step 810, the mobile explorer responds to the activation by displaying one or more dialog boxes that show a list of virtual hard drive boxes already configured for the particular terminal 300 being worked with by the user. In step 820, using the displayed dialog, the user may update account information, username information, box name information, or password information.

In step 830, the user may also use the dialog to enter the limit for the amounts of local space that is to be used for the virtual hard drive (i.e., the user box). In step 840, the user

initiates an update operation and in step 850 the mobile explorer responds by storing the options selected by the user. The options selected by the user are verified for validity through a communication between the virtual hard drive agent on the terminal and the server system manager module 110 on the server.

Fig. 9 shows an operation for adding a new box.

In step 900, the user activates an add box operation. In step 910, the mobile explorer responds with a dialog showing the current box preferences. The box preferences for mission may include account information, username information, box name information, password information, local route information, a value indicating whether the password is to be remembered, and information indicating the size for the virtual hard drive in the local storage.

In step 920, the user completes and/or updates all necessary information and initiates an update operation by clicking OK or the like. In step 930, the mobile explorer stores the information and returns to the split window view as shown, for example, in Fig. 6 with the new box visible in the mobile explorer tree area 510.

Synchronization.

Fig. 10 shows a synchronization operation. A synchronization operation may be triggered by a user initiation, such as clicking the synchronization button.

In Fig. 10, steps 1000 and 1010 take place prior to the first activation of the synchronization operation by the user. In step 1000, a directory structure is created at the root node of the mapped virtual hard drive. In step 1010, the mobile explorer refreshes the display.

In step 1020, the user activates a synchronization operation. In step 1030, the server passes to the mobile explorer the existing directory structure of the virtual hard drive as it exists at the server.

In steps 1040, 1050, 1060, and 1070 the mobile explorer inspects the server directory structure provided by the server 100 and compares it with the local directory structure existing at the terminal 300. If there are inconsistencies between the two, then the mobile explorer will instruct the server to delete or to create directories in order to bring the two directory structures in line with each other (i.e., into a consistent state).

When directories are present on the server, but not on the client, then the mobile explorer stores this information. The server, as part of the synchronization operation, will create on its own storage directories to accommodate files from the mobile explorer as necessary.

Put operation.

A put operation is described in Fig. 11. In step 1100, a user selects a file from the list in the explorer file list area. In step 1110, the user activates a put operation by

clicking on the put button. The mobile explorer responds by adding some indication to the file list area that the particular file is marked for a put operation.

In a preferred embodiment, the user is permitted to select a plurality of files for put operation before actually causing the put operation to be carried out. In an alternate embodiment (not shown), the put operation could begin as soon as the user clicks on the put button. In Fig. 11, however, a decision is made in step 1120 as to whether more files will be selected. If there are more files to be selected, then processing continues along the path marked "yes" to step 1100. If there are no more files to be selected, then processing continues along the path marked "no" to step 1130.

In step 1130, the user initiates a synchronization operation by clicking the synchronization button. In steps 1140, 1150, 1160, and 1170, the mobile explorer compares the write time of the file as kept on the terminal with the write time of the file as kept on the server. When the terminal write time it is later than the server write time, processing continues along the path marked "yes" to step 1160. When the terminal write time it is not later than the server write time, processing continues along the path marked "no" to step 1170.

In step 1160, a copy of the file is sent from the terminal 300 to the server 100. Afterward, processing continues with step 1170.

In step 1170, the file that had been marked for the put operation is deleted from the local storage of the terminal 300 and the explorer file list area is updated to show that the file has a status of "remote."

If multiple files were selected for the put operation, then each one will be processed. This is represented by the path marked "next file" between steps 1170 and 1140.

Get operation.

Fig. 12 shows a flow diagram of a get operation. In step 1200, the user selects a file. In step 1210, the user indicates a get operation is desired by clicking the get button. In step 1220, a determination is made as to whether more files are to be indicated for a get operation. If there are more files, the user executes step 1200 again (see path marked "yes"). If there are no more files, the user initiates a synchronization operation in step 1230. In step 1240, a copy of each file that has been indicated for the get operation is sent from the server to the terminal.

When a copy of a file is sent from the server to the terminal, and there already is a file of the same name at the terminal, it may be preferable to present the user with an option of copying over the file, thus over writing the local copy, or of not copying over the file.

Overall synchronization operation.

Fig. 13 shows the overall operation of a synchronization. Synchronization has been mentioned already with respect to Figs.

10-12. This description will now summarize the operations that may take place when synchronization occurs.

Synchronization, in the preferred embodiment, is an action that has multiple facets. In one sense, synchronization involves the synchronization of the directory structure of the terminal with the directory structure of the server. This synchronization is shown in Fig. 10, and is reflected in steps 1310-1330 of Fig. 13. The other sense in which synchronization is used is that of Figs. 11 and 12, in which synchronization is an action that triggers the putting and getting of files marked by a user. That is to say, put and get requests made by the user are accumulated but not executed until the occurrence of a synchronization operation. Steps 1340 and 1350 relate to the put and get operations.

Synchronization also has yet another meaning. Suppose a user is working on a file and would like to be sure that a current version of the file is kept on the server. One way to do this would be to close the file, marked the file for a put operation, and then initiate a synchronization so as to force a copy of the local file to be put to the server. This is inconvenient when the user wants to keep working on a file, because once the file is put to the server, the copy of it is deleted from the terminal.

Thus, in the preferred embodiment, synchronization also provides for the most recent version of each file to be stored at both the server and terminal. A user working on a file could

save a local copy of it and then initiate a synchronization operation. As shown in Fig. 13, at step 1360, for each file on the terminal the newest version is identified and the location having the older version (i.e., the terminal or the server) has its copy overwritten. Since, in the present example, the user is working on the file, the terminal will have the newest copy and this copy will be copied to the server, thus writing over the server version of the file while still leaving on the terminal a copy of the file for the continued use of the user.

Fig. 13 shows this overall view of synchronization, and provides for the automatic synchronization of every file. It is also possible to provide a file attributes which indicates, for each file, whether the file is to be synchronized during automatic synchronization. Thus, for files that do not change much, the synchronization attribute would be set to a value that indicates that no automatic synchronization should occur.

Memory management.

The memory of most terminals 300 is quite restricted. Memory management is necessary to help overcome this problem in an automatic and convenient manner.

Memory management according to embodiments of the invention has already been mentioned in general above, and the get operation has also been discussed generally above with respect to Fig. 12. Memory management for a get operation will now be discussed with respect to Fig. 14.

In Fig. 14, at step 1400, it is assumed that the user has picked one or more files and initiated a synchronization operation. At step 1410, the size of the present file indicated for a get operation is inspected. At step 1420, the available space on the terminal is determined. A comparison between the two is made at step 1430.

At step 1430, a determination is made as to whether the file size of the file to be retrieved is greater than the available space on the terminal. When this determination is in the affirmative, processing continues along the path marked "yes" to step 1440. At step 1440, a put operation is performed with respect to the least recently used local file.

Although the memory management scheme here moves the least recently used file from local storage to remote storage, it is also possible to select files on a different basis. For example, it is foreseen that the put operation may be taken with respect to the largest file that has not been used within a predetermined period of time.

After the put operation with respect to the least recently used local file, processing continues back with step 1420 at which point the available space on the terminal is re-evaluated. When the determination at step 1430 is in the negative, processing continues along the path marked "no" to step 1450. At step 1450, the determination is that the file size is not greater than the available space and therefore the file

indicated for the get operation is moved from the server to the terminal.

At step 1460, a determination as to whether there are any more files to get is made. When this determination is in the affirmative, processing proceeds along the path marked "yes" to step 1410. When this determination is in the negative, processing proceeds along the path marked "no" to step 1470. At step 1470, processing for the get operation comes to an end.

It should be noted that, when step 1440 is executed and a put operation is performed with respect to the least recently used local file, the status of that file is changed from "local" to "remote".

As already mentioned, the put operation does not require such careful memory management because there is always sufficient room on the server to accommodate any file from the terminal.

Security.

Different levels of security are made available to the user. It will be appreciated that higher level security requires a greater investment in processing, and that this may be inconvenient for a user whose terminal is a handheld computer.

At a basic level, security is afforded by a userid and password. Transmissions are never sent in the clear.

At a higher level of security, each transmission is arranged so that it is difficult to discern, for one

intercepting the transmission, where data begins and ends. This level of security is not encryption as usually understood, but does make it a non-trivial task for an undesired party to a communication to determine the actual content of messages.

At the highest level of security, encryption is employed. In the presently preferred embodiment, Rijndael encryption is used. When the user selects this level of security, the virtual hard drive on the mobile explorer may be shown with a padlock or similar icon adjacent thereto.

File deletion from terminal.

File deletion poses special considerations during synchronization processing. Assume that a given file, DocX, is managed by the data parking service. In other words, for a given terminal 300, a virtual hard drive has been set up and DocX has been placed on the virtual hard drive. Assume that the user has performed a get operation on DocX. In this situation, the copy of DocX is available on the terminal 300 and can be viewed in the mobile explorer file listing area 520. In the mobile explorer file listing area 520, the status shows "local". Of course, even though the file is considered to be local to the terminal 300, it is still the case that a copy of the file remains on the server 100.

Now assume that the user deletes the file at the terminal 300. During any subsequent synchronization operation, at the time that files are synchronized to ensure the most recent version is available at both the terminal 300 and the server

100, it will be discovered that there is a file on the server 100 with no counterpart on the terminal 300.

There is a possibility that the user intended to delete the file permanently from all locations, and there is an equal possibility that the user intended only to delete the file from the local terminal to free up space, but did not desire to delete the file from the system entirely.

The situation in which a user deletes a local copy of a file such as DocX is ambiguous because two versions of the file (a terminal version and a server version) exist as soon as a get operation is performed.

The ambiguity may be handled in various ways. In one embodiment, the mobile explorer toolbar 530 and menus are provided with two forms of the delete command. One form is for local deletion only, and the other form (named, for example, "purge") is for deletion at the terminal and server.

In another embodiment, the user can associate with all or any of the files a "backup" attribute. When the backup attribute is set, the local deletion of a file does not result in the deletion of the file also at the server. When the backup operation is not set, the local deletion of a file is taken to mean that the server version should also be deleted.

A flow diagram for this operation is shown in Fig. 15. In step 1500, a user deletes a local copy of a file that is being managed by the terminal virtual hard drive agent 305-2. At step 1510, a determination is made as to whether the backup attribute

applies to the file deleted. When the determination is in the affirmative, processing continues along the path marked "yes" to step 1530, the end. When the determination is in the negative, processing continues along the path marked "no" to step 1520.

At step 1520, since the backup attribute does not apply to the file deleted, it is understood that the file is to be deleted also from the server. In one embodiment, a deletion request is immediately sent to the server.

Above, however, the description has been generally given with respect to an embodiment in which put and get operations are requested by a user, but not actually carried out until the activation of a synchronization operation. Such an approach is adopted in step 1520 as shown. In step 1520, when a file is to be deleted at the server, a request for deletion of the file is created but stored at the terminal 300 until the next synchronization operation. At the next synchronization operation, the request is sent to the server. This may be accomplished, for example, at step 1330 in Fig. 13.

Conclusion.

Although various embodiments have been described, above, in a detailed manner, it will be appreciated that the details are provided simply to enable the understanding of the invention and are not intended to provide any limitation whatever to the spirit and scope of the claims. Each of the drawing figures is for explanatory purposes only, and it will be understood that

the steps may often be rearranged and performed in a different order, or in parallel where appropriate.

The invention may be manifested in different forms, as will now be described below.

Computer systems

One embodiment of this invention resides in a computer system. Here, the term "computer system" is to be understood to include at least a memory and a processor. In general, the memory will store, at one time or another, at least portions of an executable program code, and the processor will execute one or more of the instructions included in that executable program code. It will be appreciated that the term "executable program code" and the term "software" mean substantially the same thing for the purposes of this description. It is not necessary to the practice of this invention that the memory and the processor be physically located in the same place. That is to say, it is foreseen that the processor and the memory might be in different physical pieces of equipment or even in geographically distinct locations.

Computer program products

The above-identified invention may be embodied in a computer program product, as will now be explained.

On a practical level, the software that enables the computer system to perform the operations described further below in detail, may be supplied on any one of a variety of media. Furthermore, the actual implementation of the approach

and operations of the invention are actually statements written in a programming language. Such programming language statements, when executed by a computer, cause the computer to act in accordance with the particular content of the statements. Furthermore, the software that enables a computer system to act in accordance with the invention may be provided in any number of forms including, but not limited to, original source code, assembly code, object code, machine language, compressed or encrypted versions of the foregoing, and any and all equivalents.

One of skill in the art will appreciate that "media", or "computer-readable media", as used here, may include a diskette, a tape, a compact disc, an integrated circuit, a ROM, a CD, a cartridge, a remote transmission via a communications circuit, or any other similar medium useable by computers. For example, to supply software for enabling a computer system to operate in accordance with the invention, the supplier might provide a diskette or might transmit the software in some form via satellite transmission, via a direct telephone link, or via the Internet. Thus, the term, "computer readable medium" is intended to include all of the foregoing and any other medium by which software may be provided to a computer.

Although the enabling software might be "written on" a diskette, "stored in" an integrated circuit, or "carried over" a communications circuit, it will be appreciated that, for the purposes of this application, the computer usable medium will be

referred to as "bearing" the software. Thus, the term "bearing" is intended to encompass the above and all equivalent ways in which software is associated with a computer usable medium.

For the sake of simplicity, therefore, the term "program product" is thus used to refer to a computer useable medium, as defined above, which bears in any form of software to enable a computer system to operate according to the above-identified invention.

Thus, the invention is also embodied in a program product bearing software which enables a computer to perform managed data parking operations according to the invention.

User interfaces

The invention is also embodied in a user interface invocable by an application program. A user interface may be understood to mean any hardware, software, or combination of hardware and software that allows a user to interact with a computer system. For the purposes of this discussion, a user interface will be understood to include one or more user interface objects. User interface objects may include display regions, user activatable regions, and the like.

As is well understood, a display region is a region of a user interface which displays information to the user. A user activatable region is a region of a user interface, such as a button or a menu, which allows the user to take some action with respect to the user interface.

A user interface may be invoked by an application program. When an application program invokes a user interface, it is typically for the purpose of interacting with a user. It is not necessary, however, for the purposes of this invention, that an actual user ever interact with the user interface. It is also not necessary, for the purposes of this invention, that the interaction with the user interface be performed by an actual user. That is to say, it is foreseen that the user interface may have interaction with another program, such as a program created using macro programming language statements that simulate the actions of a user with respect to the user interface.

Applications programs

An application program may be several separate programs, only one program, a module of a program, or even a particular task of a module.

An applications program may be written by an applications programmer. Applications programmers develop applications programs using any of a number of programming languages. During development and design of applications programs, applications programmers may adhere to a programming methodology. A programming methodology is a set of principles by which analysis is performed and by which design decisions are made. Programming methodologies may be referred to as programming paradigms. Examples of widely-known programming paradigms

include the top-down, the data-driven, and the object oriented (OO) programming paradigms.

The object model

The OO paradigm is based on the object model. One of skill in the art readily understands the object model. For detailed information concerning the object model, a useful book, which herein is incorporated in its entirety by reference, is "Object-oriented Analysis and Design", by Grady Booch (Addison-Wesley Publishing Company).

Object oriented analysis and design (OOAD) and object oriented programming (OOP) have been the focus of great attention. OOAD and OOP are thought to provide advantages with respect to abstraction, encapsulation, modularity, and hierarchy. Furthermore, OOAD is thought to provide for improved software reuse and better adaptability to change.

According to the object model, a software system is modeled as collections of cooperating objects. Individual objects are treated as instances of a particular class. Each class has a place within a hierarchy of classes.

An object is understood to have a unique identity, to have a state, and to exhibit behavior. The behavior of an object relates to the set of operations that may be performed by the object. Such operations are also known, interchangeably, as methods of the object or as member functions of the object.

Member functions of an object are invoked by passing the object an appropriate message.

The invoking of member functions of objects to perform tasks is a central concept of the OO paradigm.

An object cannot be considered without regard to its class. Every object, when constructed, receives its structure and behavior from its class. An object may be referred to as a class instance, or as an instance of a class. Classes, in the object model, may be hierarchically related. In particular, the relationship between two classes may be a subclass/superclass relationship. A subclass may inherit the structural and behavioral features of its superclass.

Thus, whenever an object is constructed, it receives important attributes from its class. If that class is a subclass of a particular superclass, the object may receive certain attributes from the superclass as well.

Class libraries

Classes, on a practical level, may be supplied in class libraries on any one of a variety of media. Class libraries may be understood to be a kind of software. Thus, the class definitions contained in class libraries also are actually statements written in a programming language that, when executed by a computer, cause the computer to act in accordance with the particular content of the statements. Furthermore, a class library may be provided in any number of forms including, but not limited to, original source code, assembly code, object code, machine language, compressed or encrypted versions of the foregoing, and any and all computer readable equivalents.

One of skill in the art will therefore appreciate that a class library may be embodied in a computer program product as that term has already been defined, above.

Thus, the invention may be embodied in a computer program product having a class library, for example, that defines a software agent set as described herein.

THERE IS CLAIMED:

1. A method of storage management in a network over which a client and server are in communication, comprising:

defining a virtual drive for managing a plurality of files, said virtual drive definition including client storage on said client for local files and server storage on said server for remote files, wherein said plurality of files of said virtual hard drive constitute a set of managed files;

sending a get request from said client to said server identifying a given one of said remote files for a get operation;

receiving from said server file size information relating to said given one of said remote files;

making an available storage space determination based on an amount of available storage space of said client storage and said file size information; and

when said available storage space determination indicates said amount of available storage space is insufficient, and when said client has one or more of said local files, automatically:

selecting one of said one or more local files, and

performing a put operation for said selected one of said one or more local files, wherein said selected file is stored as a remote file at said server and deleted from said client.

2. The method of storage management as set forth in claim 1, wherein said automatic selecting step is performed so as to select the least recently used one of said one or more local files.

3. The method of storage management as set forth in claim 1, further comprising repetitively carrying out said selecting and said performing steps until said available storage space determination indicates said amount of available storage space is sufficient, and then storing at said client, as one of said local files, a copy of said remote file identified for said get operation.

4. The method of storage management as set forth in claim 1, further comprising:

providing a graphical user interface having:

a file list display of said managed files;

a respective user activatable file region, for each of said managed files, for indicating a user selection with respect to a corresponding one of said managed files;

a user activatable region for indicating put and get operations with respect to one or more selected ones of said managed files;

responding to an indication of said get operation by generating said get request for said one or more selected ones of said managed files.

5. The method of storage management as set forth in claim 4, further comprising:

responding to an indication of said put operation by generating a put request for said one or more selected ones of said managed files;

after generating said put request, storing said one or more selected ones of said managed files as corresponding remote files at said server and deleting said one or more selected ones of said managed files from said client.

6. The method of storage management as set forth in claim 5, further comprising:

providing said user interface with a user activatable region for indicating a synchronization operation;

responding to an indication of said synchronization operation by performing, for each of said local files, the steps of:

obtaining respective remote file write time information;

making a comparison between said respective remote file write time information and corresponding local file write time information; and

when said respective remote file write time information is earlier than said corresponding local file write time information, storing a copy of said local file said at said server.

7. The method of storage management as set forth in claim 6, further comprising:

providing said user interface with a user activatable region for indicating a local file deletion operation with respect to said one or more selected ones of said managed files;

responding to an indication of said local file deletion operation by performing, for each of said selected local files, the steps of:

determining whether a backup attribute indicates a backup status for said selected local file;

when said backup attribute indicates a backup status, deleting said selected local file without deleting a corresponding remote file from said server; and

when said backup attribute indicates no backup status, deleting said selected local file and also deleting said corresponding remote file from said server.

8. The method of storage management as set forth in claim 7, further comprising determining whether a time scheduling attribute is set for one or more of said get, put, or synchronize operations, and performing said operations in accordance with scheduling information when said time scheduling attribute is set.

9. A computer program product for enabling a client data terminal to function as a client in a network system in which said client is in communication with a server over a network, the computer program product being a computer readable medium bearing instructions, the instructions defining operations for managing storage, characterized by:

defining a virtual drive for managing a plurality of files, said virtual drive definition including client storage on said client for local files and server storage on said server for remote files, wherein said plurality of files of said virtual hard drive constitute a set of managed files;

sending a get request from said client to said server identifying a given one of said remote files for a get operation;

receiving from said server file size information relating to said given one of said remote files;

making an available storage space determination based on an amount of available storage space of said client storage and said file size information; and

when said available storage space determination indicates said amount of available storage space is insufficient, and when said client has one or more of said local files, automatically:

selecting one of said one or more local files, and

performing a put operation for said selected one of said one or more local files, wherein said selected file is stored as a remote file at said server and deleted from said client.

10. The computer program product as set forth in claim 9, wherein said automatic selecting step is performed so as to select the least recently used one of said one or more local files.

11. The computer program product as set forth in claim 9, said instructions further comprising repetitively carrying out said selecting and said performing steps until said available storage space determination indicates said amount of available storage space is sufficient, and then storing at said client, as one of said local files, a copy of said remote file identified for said get operation.

12. The computer program product as set forth in claim 9, said instructions further comprising:

providing a graphical user interface having:

a file list display of said managed files;

a respective user activatable file region, for each of said managed files, for indicating a user selection with respect to a corresponding one of said managed files;

a user activatable region for indicating put and get operations with respect to one or more selected ones of said managed files;

responding to an indication of said get operation by generating said get request for said one or more selected ones of said managed files.

13. The computer program product as set forth in claim 12, said instructions further comprising:

responding to an indication of said put operation by generating a put request for said one or more selected ones of said managed files;

after generating said put request, storing said one or more selected ones of said managed files as corresponding remote files at said server and deleting said one or more selected ones of said managed files from said client.

14. The computer program product as set forth in claim 13, said instructions further comprising:

providing said user interface with a user activatable region for indicating a synchronization operation;

responding to an indication of said synchronization operation by performing, for each of said local files, the steps of:

obtaining respective remote file write time information;

making a comparison between said respective remote file write time information and corresponding local file write time information; and

when said respective remote file write time information is earlier than said corresponding local file write time information, storing a copy of said local file said at said server.

15. The computer program product as set forth in claim 14, said instructions further comprising:

providing said user interface with a user activatable region for indicating a local file deletion operation with respect to said one or more selected ones of said managed files;

responding to an indication of said local file deletion operation by performing, for each of said selected local files, the steps of:

determining whether a backup attribute indicates a backup status for said selected local file;

when said backup attribute indicates a backup status, deleting said selected local file without deleting a corresponding remote file from said server; and

when said backup attribute indicates no backup status, deleting said selected local file and also deleting said corresponding remote file from said server.

16. The computer program product as set forth in claim 15, said instructions further comprising determining whether a time scheduling attribute is set for one or more of said get, put, or synchronize operations, and performing said operations in

accordance with scheduling information when said time scheduling attribute is set.

17. A program product for enabling a client data terminal to function as a client in a network system in which said client is in communication with a server over a network and to participate in a system for managing storage, the computer program product being a computer readable medium bearing a class library, the class library including:

a class for constructing a manager agent object, and a class for constructing a virtual hard drive agent object;

said manager agent object class defining member functions for communicating with a server and passing messages between said server and said virtual hard drive agent object;

said virtual hard drive agent object class defining member functions for:

defining a virtual drive for managing a plurality of files, said virtual drive definition including client storage on said client for local files and server storage on said server for remote files, wherein said plurality of files of said virtual hard drive constitute a set of managed files;

sending a get request from said client to said server identifying a given one of said remote files for a get operation;

receiving from said server file size information relating to said given one of said remote files;

making an available storage space determination based on an amount of available storage space of said client storage and said file size information; and

when said available storage space determination indicates said amount of available storage space is insufficient, and when said client has one or more of said local files, automatically:

selecting one of said one or more local files, and

performing a put operation for said selected one of said one or more local files, wherein said selected file is stored as a remote file at said server and deleted from said client.

18. The program product as set forth in claim 17, wherein said member function for carrying out said automatic selecting step is defined so as to select the least recently used one of said one or more local files.

19. The program product as set forth in claim 17, said member functions of said virtual hard drive agent object further comprising repetitively carrying out said selecting and said performing steps until said available storage space determination indicates said amount of available storage space is sufficient, and then storing at said client, as one of said local files, a copy of said remote file identified for said get operation.

20. The computer program product as set forth in claim 17, said member functions of said virtual hard drive agent object further comprising:

providing a graphical user interface having:

a file list display of said managed files;

a respective user activatable file region, for each of said managed files, for indicating a user selection with respect to a corresponding one of said managed files;

a user activatable region for indicating put and get operations with respect to one or more selected ones of said managed files;

responding to an indication of said get operation by generating said get request for said one or more selected ones of said managed files.

21. The computer program product as set forth in claim 20, said member functions of said virtual hard drive agent object further comprising:

responding to an indication of said put operation by generating a put request for said one or more selected ones of said managed files;

after generating said put request, storing said one or more selected ones of said managed files as corresponding remote files at said server and deleting said one or more selected ones of said managed files from said client.

22. The computer program product as set forth in claim 21, said member functions of said virtual hard drive agent object further comprising:

providing said user interface with a user activatable region for indicating a synchronization operation;

responding to an indication of said synchronization operation by performing, for each of said local files, the steps of:

obtaining respective remote file write time information;

making a comparison between said respective remote file write time information and corresponding local file write time information; and

when said respective remote file write time information is earlier than said corresponding local file write time information, storing a copy of said local file said at said server.

23. The computer program product as set forth in claim 22, said member functions of said virtual hard drive agent object further comprising:

providing said user interface with a user activatable region for indicating a local file deletion operation with respect to said one or more selected ones of said managed files;

responding to an indication of said local file deletion operation by performing, for each of said selected local files, the steps of:

determining whether a backup attribute indicates a backup status for said selected local file;

when said backup attribute indicates a backup status, deleting said selected local file without deleting a corresponding remote file from said server; and

when said backup attribute indicates no backup status, deleting said selected local file and also deleting said corresponding remote file from said server.

24. The computer program product as set forth in claim 23, said member functions of said virtual hard drive agent object further comprising determining whether a time scheduling attribute is set for one or more of said get, put, or synchronize operations, and performing said operations in accordance with scheduling information when said time scheduling attribute is set.

25. A client data terminal, comprising:

a software agent set, including:

a master agent adapted to communicate with an external system manager module of a server so as to permit said system manager test port exercise control of the agents in said software agent set through said communication with said master agent, and

a virtual hard drive agent supporting managed data parking services;

said client data terminal being free of mass storage; and

said client data terminal being adapted to store user files in a mass storage at said server at a location determined by said system manager module, said storing being under control of said software agent set by said system manager module.

26. The client data terminal as set forth in claim 25, further comprising a security agent supplying logon and subscribed services security parameters to a security system on said server; wherein said client data terminal does not receive any stored files from said server in the event that a user of said client data terminal does not provide proper authentication.

FIG. 1

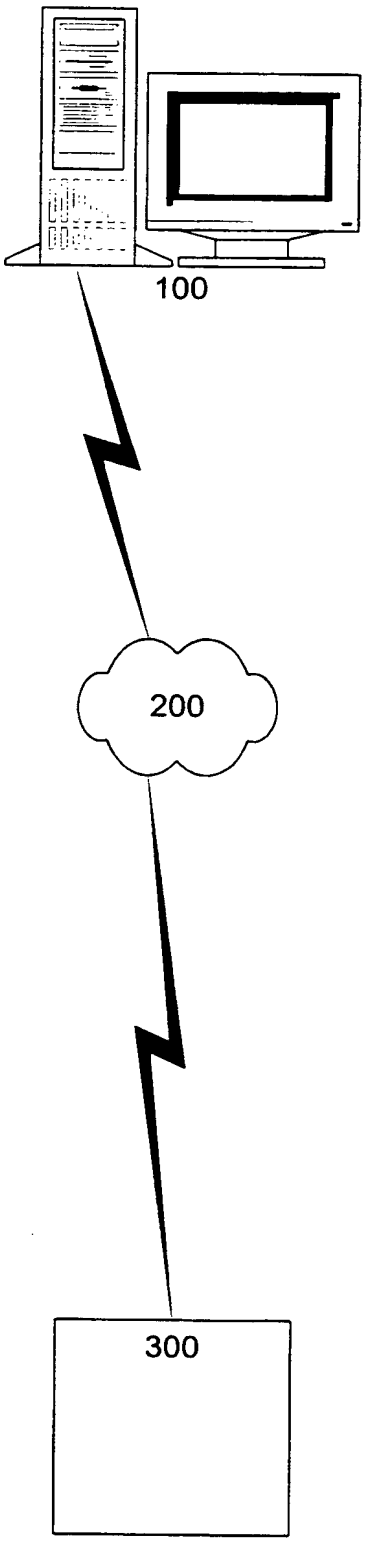


FIG. 2

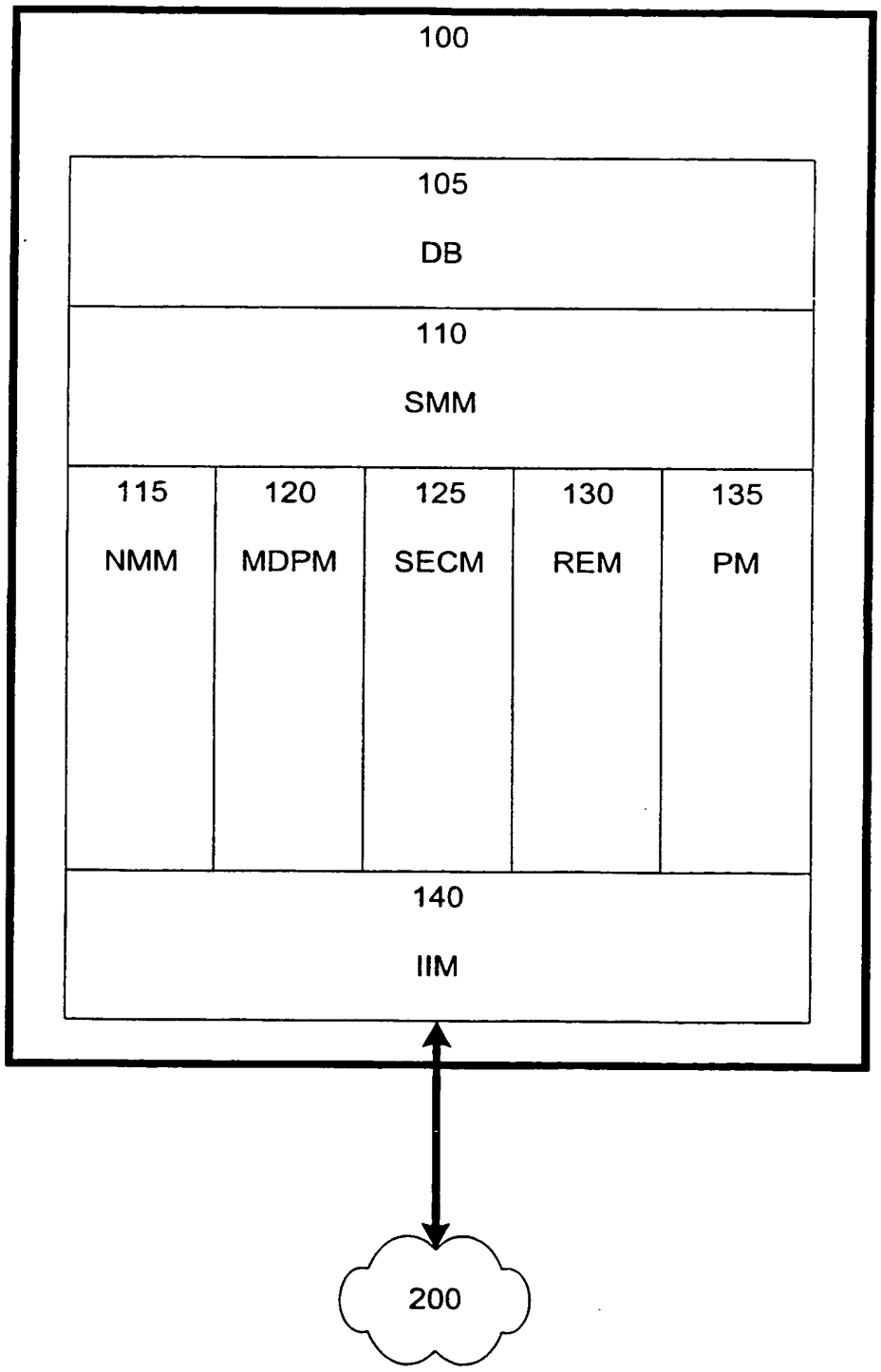


FIG. 3

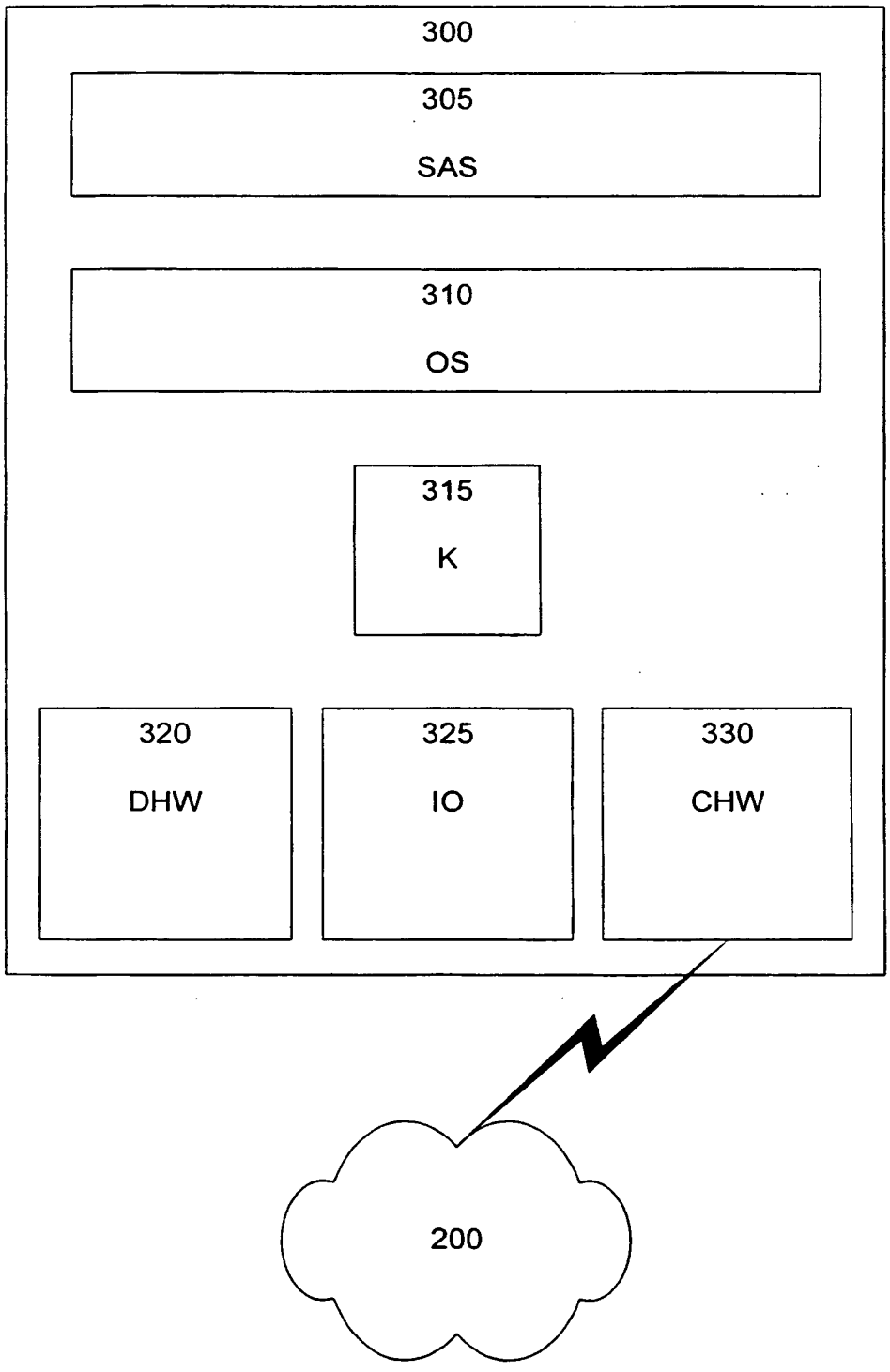


FIG. 4

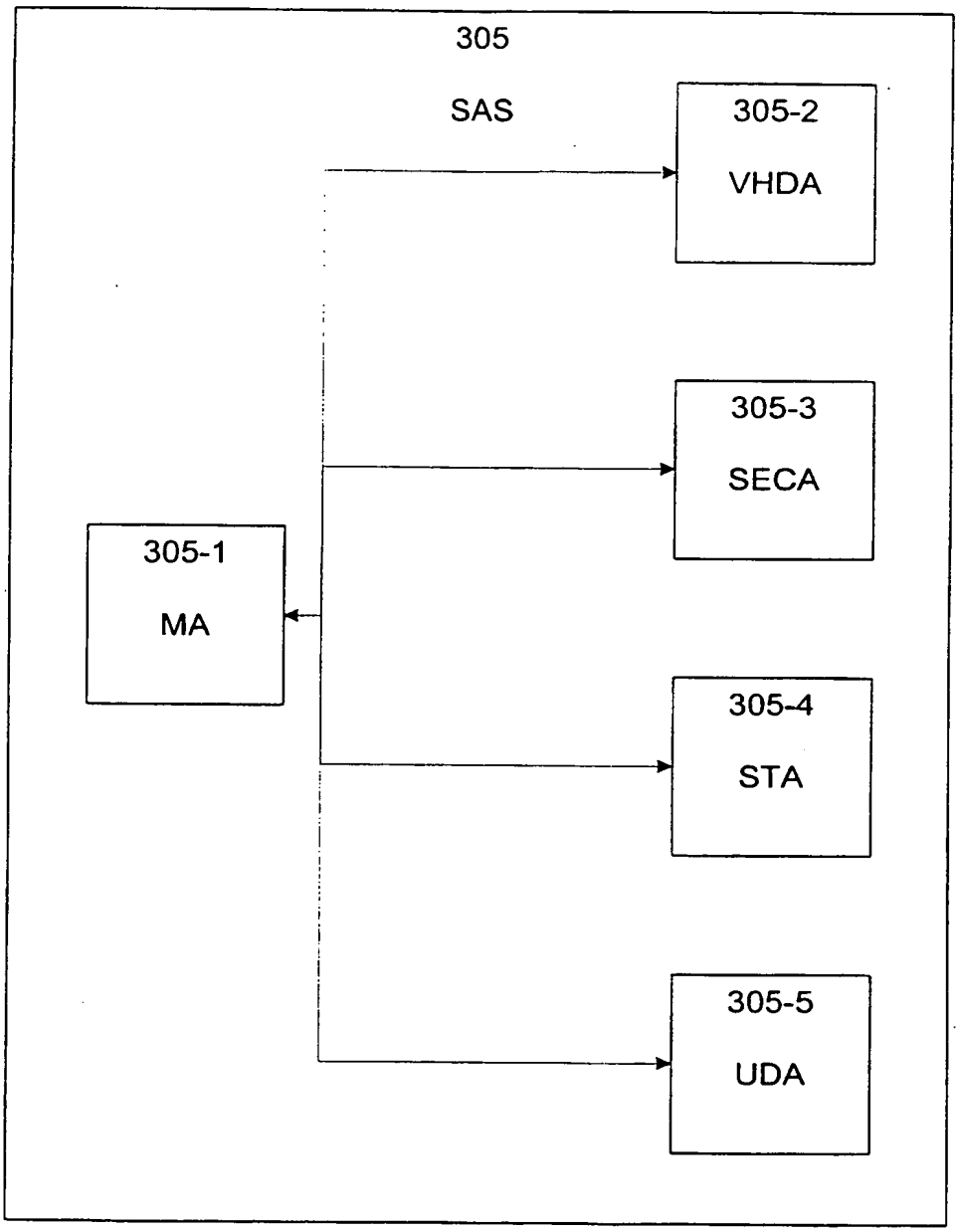


FIG. 5

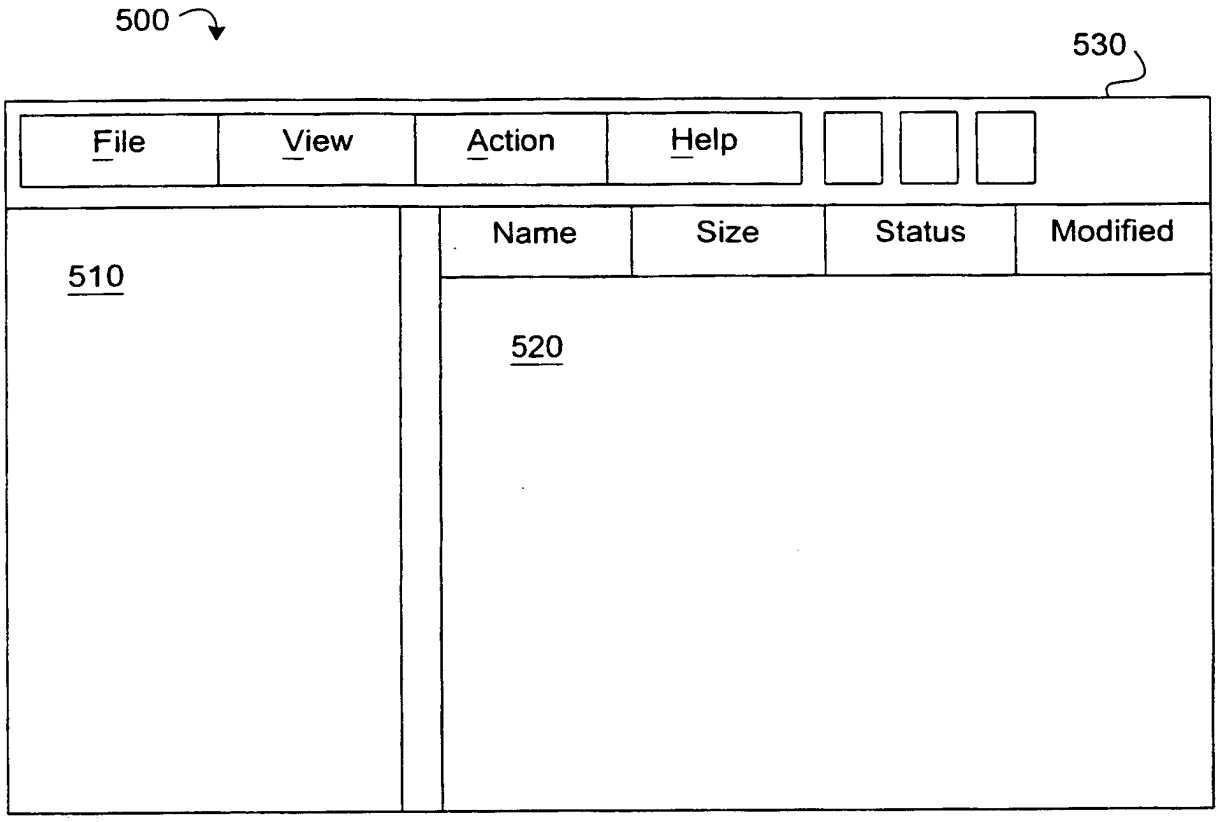


FIG. 6

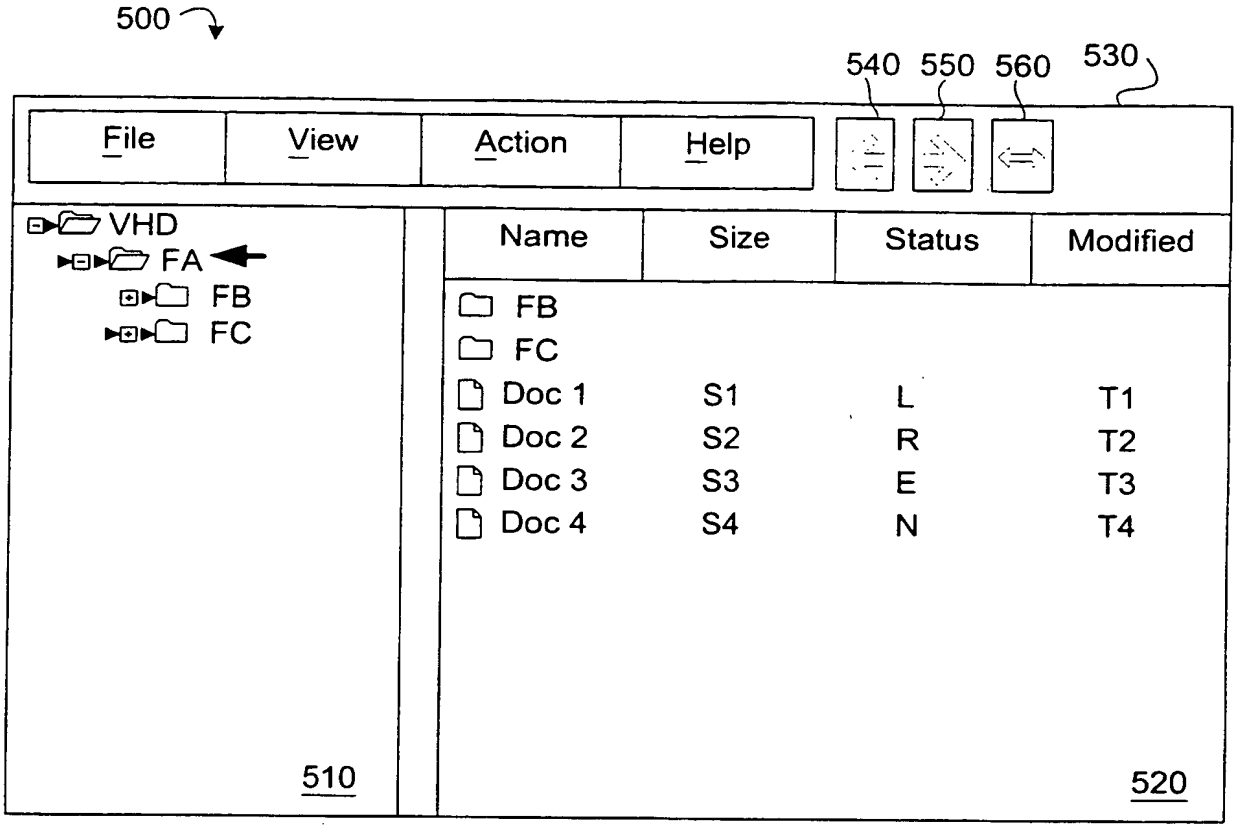


FIG. 7

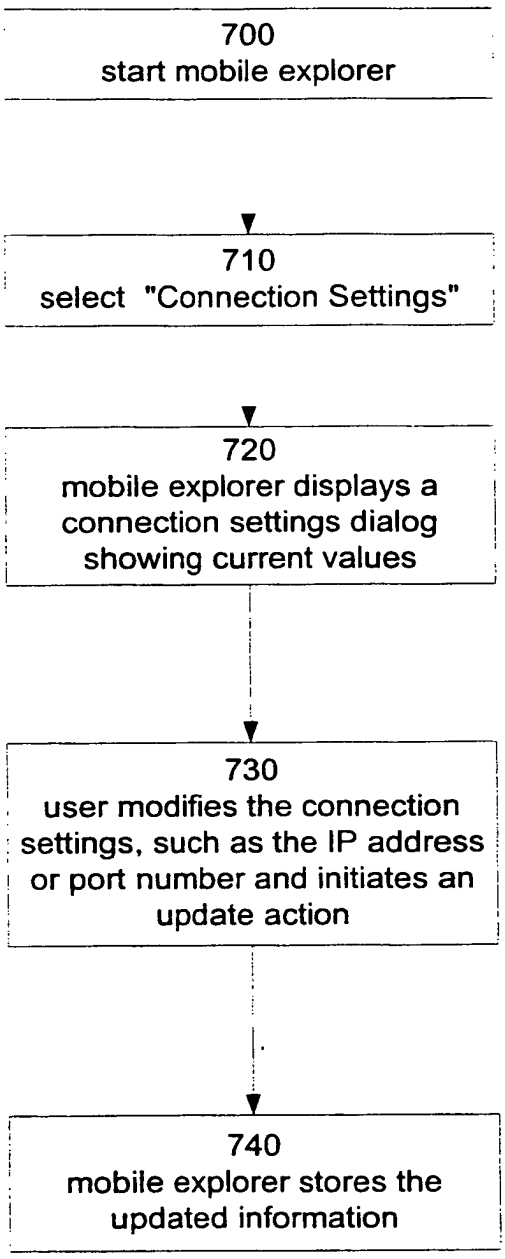


FIG. 8

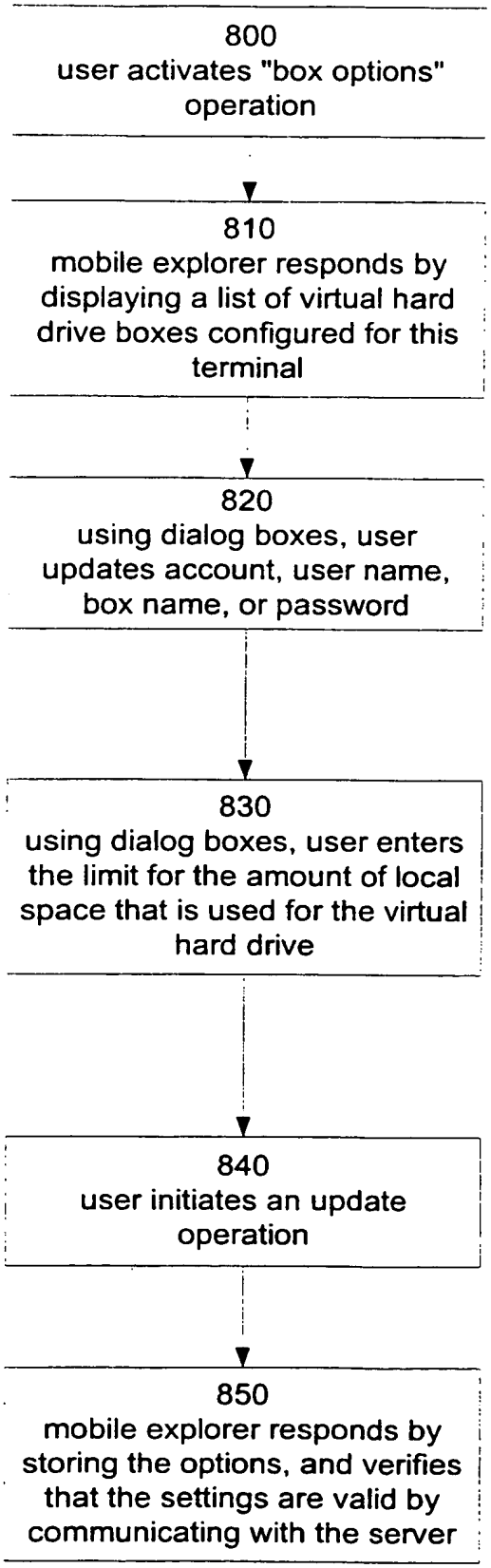


FIG. 9

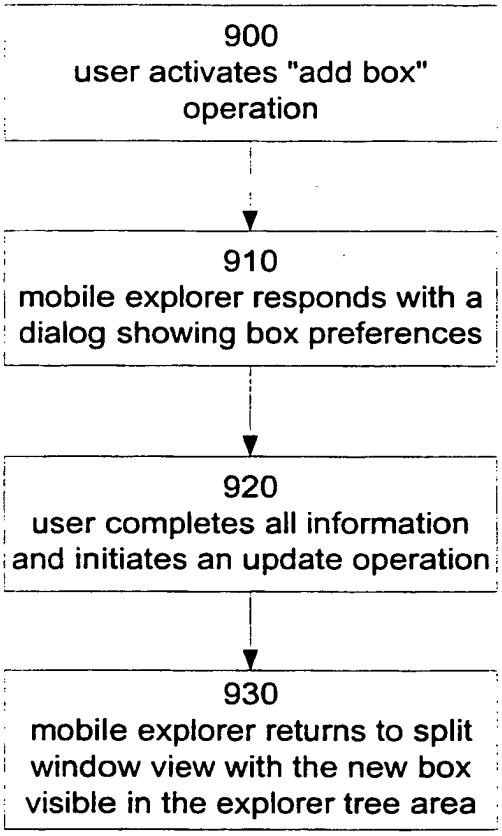


FIG. 10

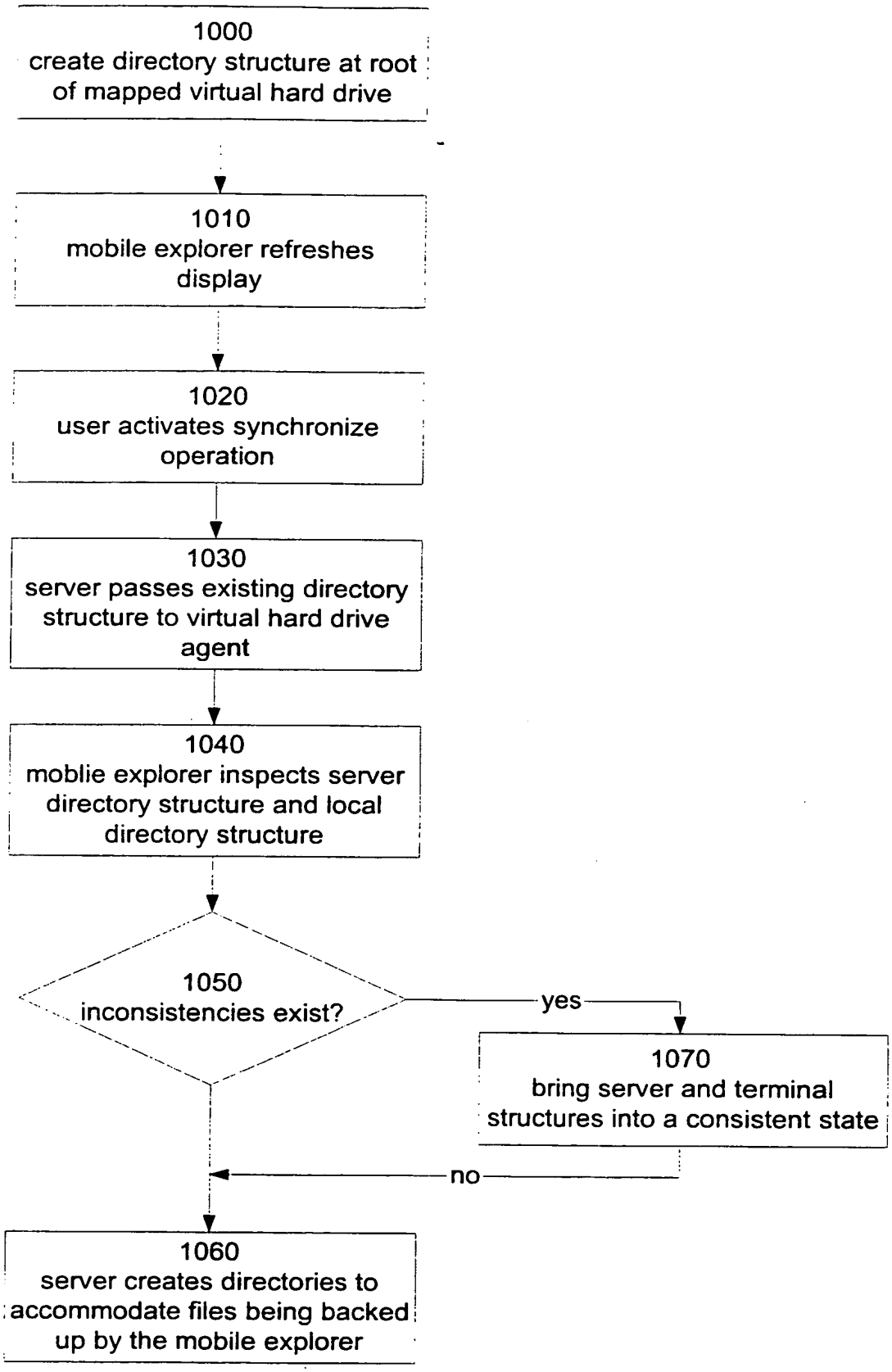


FIG. 11

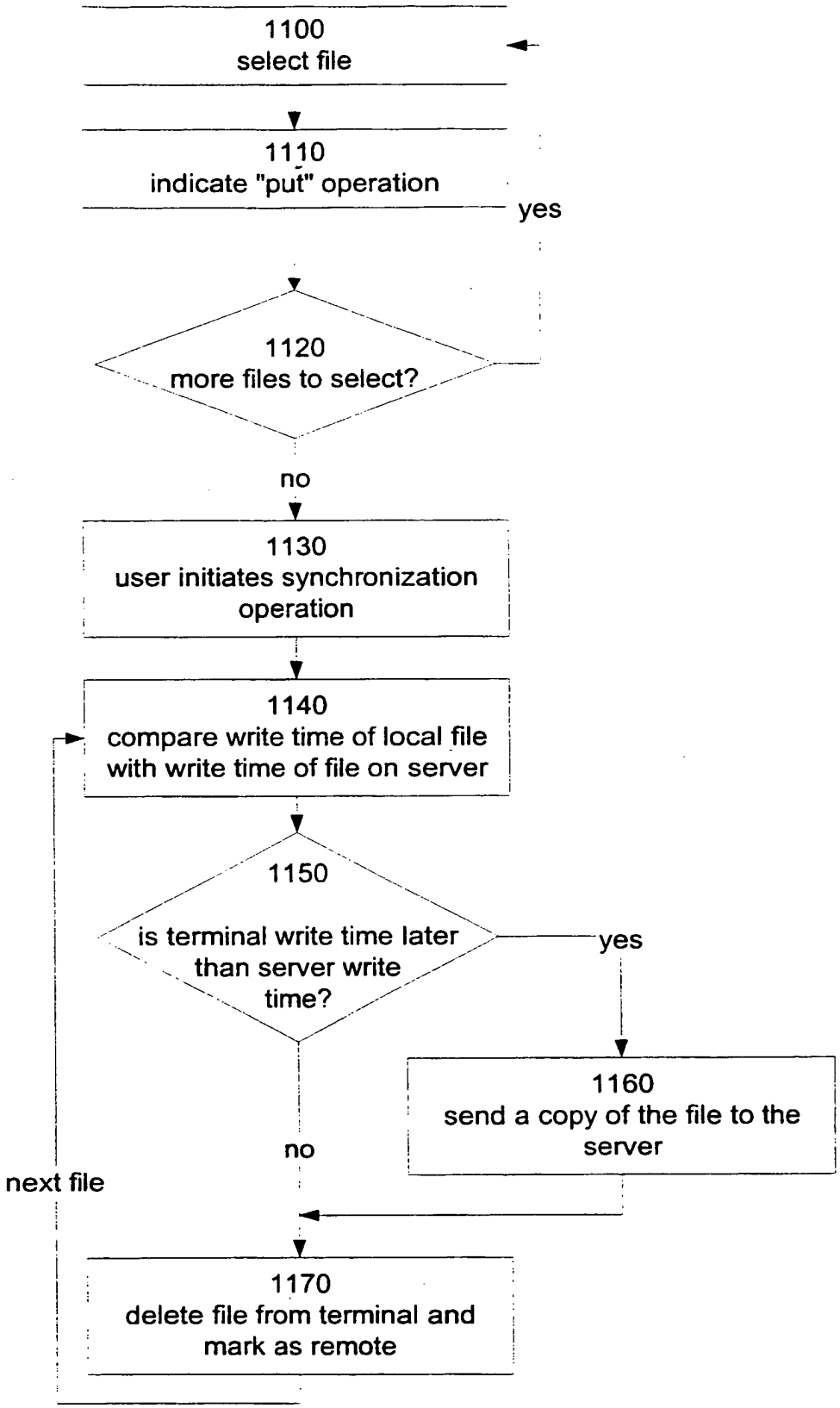

1100
select file
1110
indicate "put" operation
yes
1120
more files to select?
no
1130
user initiates synchronization operation
1140
compare write time of local file with write time of file on server
1150
is terminal write time later than server write time?
yes
1160
send a copy of the file to the server
no
next file
1170
delete file from terminal and mark as remote

FIG. 12

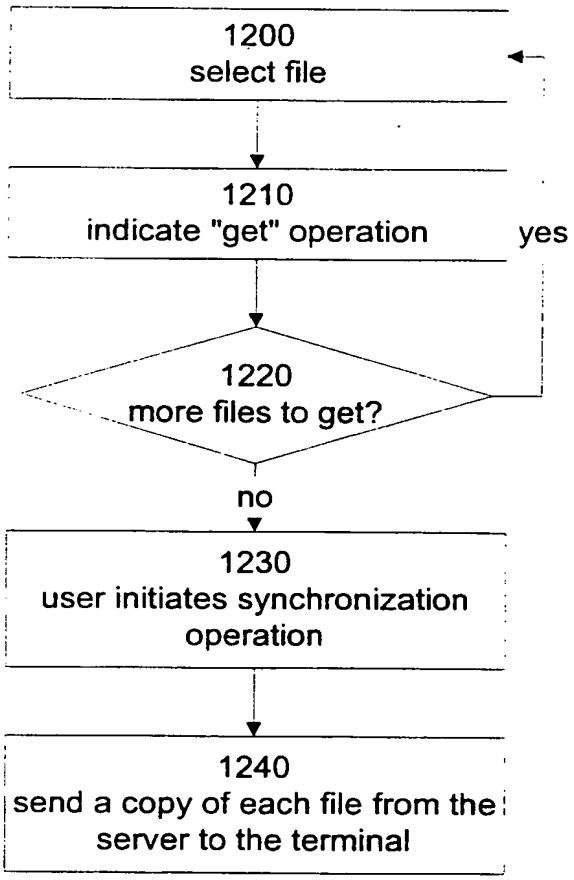


FIG. 13

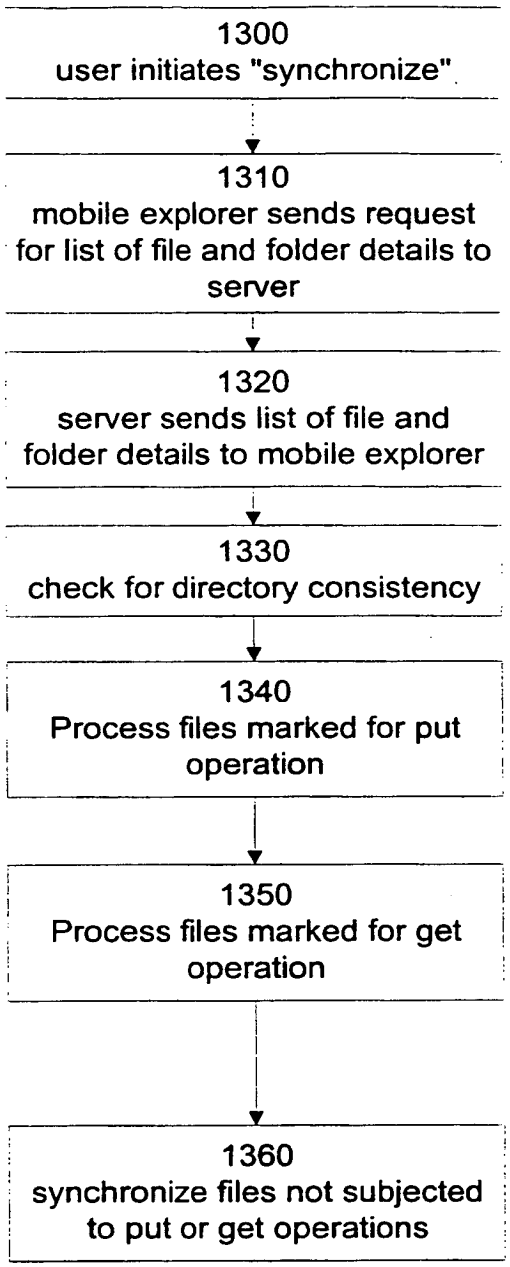


FIG. 14

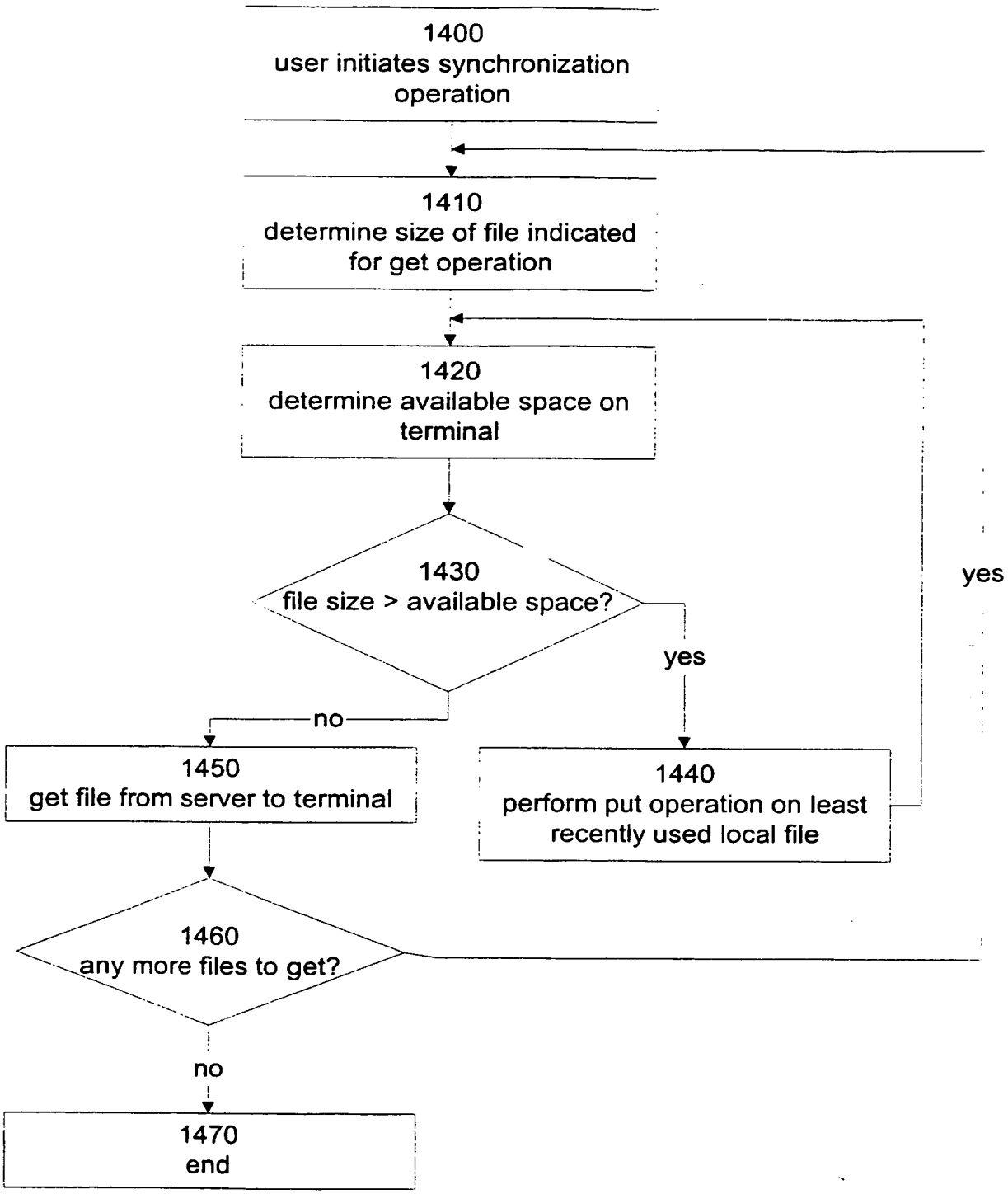
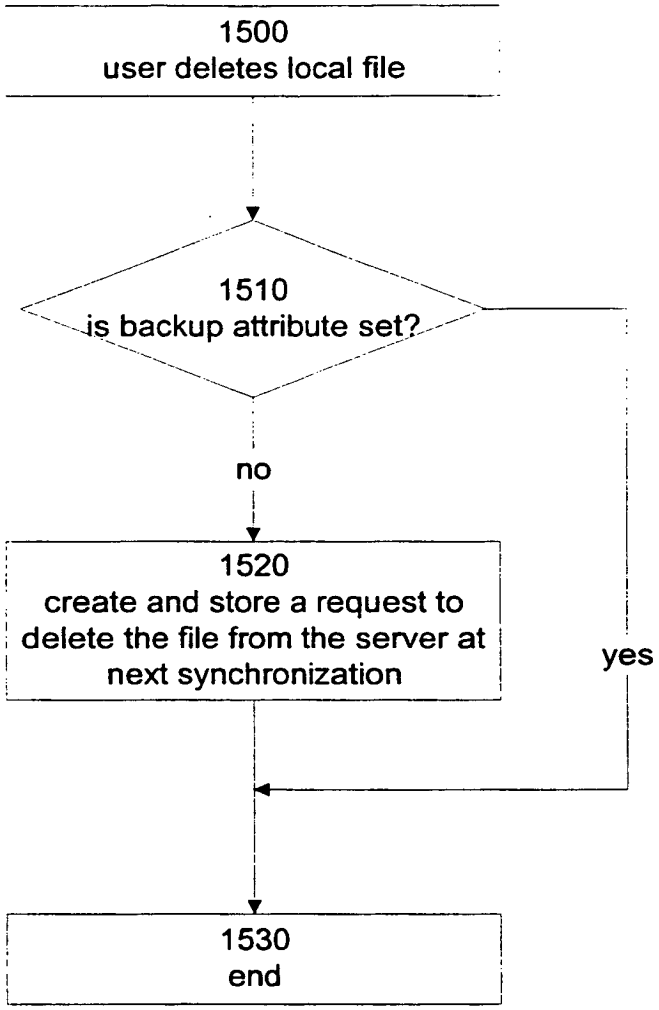


FIG. 15



INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/08744

A. CLASSIFICATION OF SUBJECT MATTER

IPC(7) : G06F 17/30

US CL : 707/201, 202, 203, 204; 711/118, 161; 706/908

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

U.S. : 707/201, 202, 203, 204; 711/118, 161; 706/908

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

Please See Extra Sheet.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,463,772 A (THOMPSON et al) 31 October 1995, col, 1, lines 9-63, col. 2, lines 24-52 and lines 66-67, col. 3, lines 1-66, col. 4, lines 14-67, col. 5, lines 37-67, col. 6, lines 1-67, col. 7, lines 1-29 and lines 40-62, col. 8, lines 14-21 and lines 32-66, col. 9, lines 16-29 and lines 40-61, col. 10, lines 14-26 and lines 60-67, col. 11, lines 1-5 and lines 57-67, col. 12, lines 1-34 and lines 61-66, col. 13, lines 33-45, col. 14, lines 28-35, col. 15, lines 34-67, col. 17, lines 36-55, col. 18, lines 59-67, col. 19, lines 1-42, col. 20, lines 1-31, col. 21, lines 12-19, col. 22, lines 4-67, col. 23, lines 1-23, col. 24, lines 43-61, col. 25, lines 22-25 and lines 45-54, col. 28, lines 28-41, and col. 31, lines 1-3.	1-26

[X] Further documents are listed in the continuation of Box C. [ ] See patent family annex.

*	Special categories of cited documents:	"T"	later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A"	document defining the general state of the art which is not considered to be of particular relevance	"X"	document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E"	earlier document published on or after the international filing date	"Y"	document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L"	document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&"	document member of the same patent family
"O"	document referring to an oral disclosure, use, exhibition or other means		
"P"	document published prior to the international filing date but later than the priority date claimed		

Date of the actual completion of the international search	Date of mailing of the international search report
07 JULY 2000	15 AUG 2000
Name and mailing address of the ISA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231	Authorized officer KIM VU
Facsimile No. (703) 305-3230	Telephone No. (703) 305-4393

Form PCT/ISA/210 (second sheet) (July 1998)*

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/08744

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 5,495,607 A (PISELLO et al) 27 February 1996, col. 3, lines 25-67, col. 4, lines 1-25 and lines 49-60, col. 6, lines 39-67, col. 7, lines 1-15 and lines 31-58, col. 8, lines 28-41, col. 11, lines 4-16, col. 12, lines 54-61, col. 15, lines 40-51, and lines 60-67, col. 16, lines 1-17, col. 19, lines 30-38, col. 22, lines 61-67, col. 23, lines 1-3, col. 29, lines 48-57.	1-26
Y	US 5,712,976 A (FALCON, Jr. et al) 27 January 1998, col. 1, lines 57-67, col. 2, lines 1-25 and lines 52-67, col. 7, lines 17-29, col. 8, lines 10-55, col. 9, lines 20-33, col. 13, lines 28-36, col. 18, lines 46-56, col. 19, lines 20-28 and lines 50-58, col. 30, lines 19-31, and col. 33, lines 12-25.	1-26

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US00/08744

B. FIELDS SEARCHED
Electronic data bases consulted (Name of data base and where practicable terms used):

WEST
Search terms: storage management, network, files, managed files, file size, local files, graphical user interface, file list, file region, synchronization.

Form PCT/ISA/210 (extra sheet) (July 1998)*